AFRL-RI-RS-TR-2017-219

DETECTION AND LEARNING OF UNEXPECTED BEHAVIORS OF SYSTEMS OF DYNAMICAL SYSTEMS BY USING THE Q2 ABSTRACTIONS

NORTHEASTERN UNIVERSITY

NOVEMBER 2017

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2017-219 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
WILLIAM D. LEWIS
Work Unit Manager

/ S /
JULIE BRICHACEK
Chief, Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
NOV 2017	FINAL TECHNICAL REPORT	May 2015 – May 2017

4. TITLE AND SUBTITLE

DETECTION AND LEARNING OF UNEXPECTED BEHAVIORS OF SYSTEMS OF DYNAMICAL SYSTEMS BY USING THE Q2 ABSTRACTIONS

5a. CONTRACT NUMBER
FA8750-15-1-0095

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Mitch Kokar, Shweta Singh, Shan Lu

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
R1MP

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Northeastern University
360 Huntington Avenue
Boston MA 02115

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RISC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2017-219

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report describes the research on characterization and detection of emergent behaviors in groups of dynamical systems (agents) performing a common mission. The mission was related to monitoring of plume. The main objective of this research was the reduction of the complexity of emergence detection through the use of the theory of similitude and of qualitative abstractions of dynamical systems.

15. SUBJECT TERMS

Emergence, emergent behaviors, agents, UAVs, plume monitoring, similitude theory, qualitative abstractions of dynamical systems

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**WILLIAM D. LEWIS**
U	U	U	UU	74	19b. TELEPHONE NUMBER (*Include area code*)

Table of Contents

Section **Page**

List of Figures ..... **iii**

List of Tables ..... **vi**

List of Algorithms ..... **vi**

1 SUMMARY **1**

2 INTRODUCTION **2**
2.1 Emergence/Emergent Properties 2
2.2 On Irreducibility in Mathematics 6
2.3 Computational Irreducibility 7
2.4 Classification/Taxonomy of Emergent Behaviors 9
2.4.1 Fromm's Taxonomy 11
2.5 Emergence in Swarms of UAVs 12

3 METHODS, ASSUMPTIONS AND PROCEDURE **13**
3.1 Targeting Emergence: EBS (Emergent Behavior System) Framework 13
3.1.1 Emergence in Levels of the System. 13
3.1.1.1 Agent. 15
3.1.1.2 Interaction between Agents. 15
3.1.1.3 Agent Environment. 16
3.1.1.4 System. 16
3.1.1.5 External Entity - Commander (Observer). 17
3.1.2 Emergence in Ontology 17
3.1.2.1 Ontology Basics. 17
3.1.2.2 Features of Emergent Behaviors. 17
3.1.3 EBS Simulation Framework 19
3.1.3.1 Behavior Model 19
3.1.3.2 Behavior Classifier. 20
3.1.3.3 Simulation Engine. 21
3.1.3.4 Visualizer 22
3.1.3.5 Behavior Monitor. 22
3.2 Scenario Formalization: Undesirable Emergent Behaviors in Swarms of UAVs 23
3.2.1 Problem Specification. 23
3.2.2 Agent (UAV) Dynamics. 25
3.2.3 Dimensionality Problem 26
3.3 Qualitatively Different Behaviors 27
3.4 Quantitative-Qualitative (Q^2) Approach 28

4 RESULTS AND DISCUSSION **30**

4.1 Boids Flocking 30
4.2 Agent Based Modeling and Simulation of UAVs 30
4.2.1 Single UAV representation. 30
4.2.2 Extension to multiple UAVs. 32
4.2.3 Simulating Emergent Behavior Types. 35
4.3 Application of Q^2 Approach 37
4.3.1 A Test Case Scenario. 37
4.3.2 Generalized Q^2 Conceptualization. 39
4.3.2.1 Partitioning of the Output Space and Qualitative Outputs. 39
4.3.2.2 Partitioning of State Space and Qualitative States. 40
4.3.2.3 Partitioning of Input Space and Qualitative Inputs. 43
4.3.3 Proof of consistency constraints. 45
4.3.4 Simulation in Qualitative Domain. 47
4.4 Detection and Classification of Emergent Behaviors 48
4.4.1 Detection using Variety 48
4.4.2 Detection using FSM. 49
4.4.3 Classification using Ontology Inference. 50
4.5 Learning Hypersurfaces 51
4.6 Complexity Analysis 55

5 CONCLUSION 56
5.1 Future Direction 58

APPENDIX A - Publications and Presentations 65

APPENDIX B - Abstracts 66

LIST OF ACRONYMS 67

List of Figures

Figure **Page**

1 Concept Lattice with Emergence Features and Corresponding Reference using Formal Concept Analysis 4
2 Overlap of Features between Definitions....5
3 Taxonomy of Emergent Behaviors by Fromm [1] 12
4 Relationship between Micro and Macro Levels 14
5 Agent(s) in EBS....15
6 Proposed EBS Framework Showing OWL and Simulation Based Components 19
7 Behavior Modeling Concepts Based on Nuvio....20
8 Ontology Concepts for Representing Finite State Machine.... 21
9 Main Ontological Concepts for Representing Structure of a Multi-Agent System 22
10 Classification of Emergent Behaviors 23
11 Surveillance Area (NetLogo Snapshot)....24
12 Q^2: Quantitative-Qualitative Representation of a Dynamical System 29
13 NetLogo Simulation - Boids Flocking....31
14 Detection of Flocking Behavior using Similitude Theory....31
15 UAV and Information Age 32
16 Pattern Emergence from Single UAV (Plume - Green Area) 33
17 Control Policies for Multiple UAVs (grey circle represent the $S(t)$ of center UAV.... 34
18 NetLogo Simulation of persistent surveillance of square plume by 4 UAVs....35
19 NetLogo Simulation of persistent surveillance of circular plume by 4 UAVs 36
20 Flocking Emergent Behaviors in Multi-UAV Plume Monitoring System 37
21 Undesirable Emergent Behavior - Non-Covered Facility....38
22 NetLogo Simulation of Persistent Surveillance by 4 UAVs of Square Plume (Green Region) - Minimization of Information Age (Right Plot) 39
23 NetLogo Simulation of Persistent Surveillance by 4 UAVs of Square Plume (Green Region) - Undesirable Group Formation 40
24 Two UAVs Moving in Circular Motion (Opposite Directions) 41
25 Observation of Outputs....41
26 Transition through Qualitative Space....42
27 Invariance in Outputs....42
28 Partitioning of State Space [Red: θ_1, Yellow: θ_2, Magenta: θ_3, Cyan: θ_4, Green: θ_5, Blue: θ_6, Black: θ_7, Light Blue: θ_8]....44
29 Automaton: Qualitative State Transition Function 45
30 Partitioning of Input Space [Red: λ_1, Yellow: λ_2, Magenta: λ_3, Cyan: λ_4, Green: λ_5, Blue: λ_6, Black: λ_7, Light Blue: λ_8].... 46
31 Consistency Constraint [Red: ω_1, Yellow: ω_2, Magenta: ω_3, Cyan: ω_4, Green: ω_5, Blue: ω_6, Black: ω_7, Light Blue: ω_8]....46
32 MATLAB Simulation of Type IIa (left) and Type IIb (right) Emergent Behavior using Q^2....47
33 Detection of Emergent Behavior using Variable-Based Approach (Variety Metric).....49
34 Detection using Variety with Dimensional Variables 50

35 Detection using Variety with Dimensionless Variables....51
36 Two Instances of FSM....52
37 Representation of Two FSM Instances in OWL....53
38 Two Event Trace Examples....54
39 UAV Flocking Example....54
40 Accuracy of Learning Algorithm [2]....54
41 Comparison between Prediction by Similarity and Traditional Approach....55
42 Possible Automata for Four Traces....56
43 Number of Comparisons for N Qualitative Partitions....57
44 Comparison of Number of States with Dimensional and Dimensionless Variables ($z = 12$)....58
45 General Dynamical System - GDS (Left) versus Qualitative Dynamical System - QDS (Right)....59

List of Tables

Table **Page**

1 Different Classifications and their Criteria....11

2 Features of Emergent Behaviors....18

List of Algorithms

Algorithm **Page**

1 Agent Logic....34

2 System Logic....34

3 Hypersurface Learning....53

1 SUMMARY

Many studies have shown that swarms of collaborating autonomous agents, e.g., UAVs (Unmanned Aerial Vehicles) performing a specific mission, can solve problems better than collections of agents that are controlled centrally. However, controlling a large number of such agents is not only very complicated, but also requires many humans to be part of the control loop. Additionally, studies have also shown that swarms of autonomous agents can exhibit unpredictable, and often undesirable, behaviors, termed *emergent behaviors*. Consequently, before giving control over their missions to such swarms of agents, it is necessary to establish some mechanisms to (1) detect (and if possible predict) that an undesirable behavior is imminent, and (2) provide ways to control such swarms so that such behaviors can be avoided. In the current work, we address such issues and present techniques to analyze undesirable behaviors in swarms of autonomous agents. These research efforts focus specifically on swarms of UAVs performing a specified mission. We formalized a specific scenario of persistent surveillance where the aim is to provide monitoring of the plume (targeted search area) such that the metric of "information age" is minimized.

Our approach relies on the theory of *similitude* (or physical similarity). This theory has been used extensively in physics and engineering, in particular to model behaviors of phenomena that occur due to the interactions of particles, e.g., in heat and mass exchange. We apply similar methods to the modeling and analysis of behaviors of multiple UAVs, as a whole treated as a complex dynamical system. The main idea of the similitude theory is that similar behaviors occur when the values of the system variables are in a specific relation. Such relations can be captured by the so called dimensionless quantities. Each such relation defines a hypersurface in the space spanned over the system variables. We represent such relationships canonically, and then show how any description of a hypersurface can be generated automatically. Moreover, we also show how such relationships can be learned by machine learning algorithms. Knowing such relationships will allow the system to measure the distance to potentially undesirable behaviors, alerting the central controller, which in turn can provide the agents with policies for avoiding such behaviors. The approach also uses the structure, termed the Q^2 system, that integrates a quantitative dynamical system with a qualitative dynamical system (represented by an automaton). The use of this structure lowers the computational complexity of the algorithms for detecting and predicting undesirable behaviors by two orders of magnitude with respect to a more traditional approach based on just a general quantitative dynamical system.

One of the tasks of this work is also to study the concept of emergent behavior in general (desirable or undesirable) and develop the taxonomy of such behaviors in OWL (Web Ontology Language). The ontology includes a classification of behaviors, their characteristics, as well as the relationships among the behaviors and the characteristics.

The outcomes of this research include: (1) An ontology of emergent behaviors, a taxonomy of definitions of emergence and analysis of definitions of emergence and the semantics of the terms that they use. (2) Simulations exemplifying undesirable behaviors in swarms of UAVs for the specified scenario of plume monitoring by swarms of UAVs. (3) Global control policies that result in the emergence of different types of behaviors, including both desirable and undesirable. (4) Algorithms for learning critical hypersurfaces for

partitioning the system spaces into qualitative inputs, states and outputs and for constructing qualitative state machines. (5) Results of formal analysis of the complexity and efficiency of the approach.

2 INTRODUCTION

Systems with large numbers of components and intricate interactions are pervasive, including natural systems, ranging from animal flocks [3] to socio-ecological systems [4] as well as sophisticated artificial systems such as the Internet [5], social networks [6] and large scale distributed computer systems [7]. These systems, commonly termed as Complex Adaptive Systems (CAS), may exhibit emergent behaviors due to the various non-linear spatiotemporal interactions among large numbers of components and subsystems [8]. These interactions may lead to some properties that are not derivable from the properties of individual components. These properties are often termed as *emergent properties* or *emergence*. Emergent behavior, by its name, is a behavior or pattern emergent from its constituents (or parts). Whether this "emergence" is traceable to its constituents is a question that has caused a more than two decade-long debate and resulted in several definitions of emergent behaviors [9]. Emergence makes a system harder to analyze and design, and requires a formal approach for detecting and reasoning about its causes and nature [1, 10, 11]. This section presents our literature review of emergence and its definitions. We also present a review of the different classifications/ taxonomy of emergent behaviors and emergence in swarms of UAVs.

2.1 Emergence/Emergent Properties

The term "emergence" is used in the literature to refer to a kind of behavior observed in a wide spectrum of phenomena. The term "emergent" was coined by philosopher G. H. Lewes [12] in 1875, who wrote:

> "Every resultant is either a sum or a difference of the co-operant forces; their sum, when their directions are the same – their difference, when their directions are contrary. Further, every resultant is clearly traceable in its components, because these are homogeneous and commensurable. It is otherwise with emergents, when, instead of adding measurable motion to measurable motion, or things of one kind to other individuals of their kind, there is a co-operation of things of unlike kinds. The emergent is unlike its components insofar as these are incommensurable, and it cannot be reduced to their sum or their difference."

Emergence has been studied in many disciplines, including philosophy, physics, biology, economics, computer science and engineering. Many attempts to define the meaning of this term have been documented. Each of these fields has their own understanding and interpretation of what emergence means. However, no agreed upon definition exists till now. In this section, we attempt to study and categorize different definitions that exist in literature.

Generally, the existing definitions of emergence define the concept in three categories: (1) define emergence as properties that are not localized in one component but result from the component interactions [13, 14, 15, 16]; (2) define emergence by pointing to the difficulties of predicting the emergent properties [17, 18]; (3) define emergence as the difference between the predicted and the realized design of software system, without revealing the nature of the emergent properties and their essence [19, 20]. Presented below is the small sampling of the definitions from literature that support the above mentioned three categories:

- **Chalmers (1996)**: "an interesting property that is unexpected, given the underlying principles governing the system." [21]
- **G. Dyson (1998)**: "Emergent behavior is that which cannot be predicted through analysis at any level simpler than that of the system as a whole. Explanations of emergence, like simplifications of complexity, are inherently illusory and can only be achieved by sleight of hand. This does not mean that emergence is not real. Emergent behavior, by definition, is what's left after everything else has been explained." [22]
- **Holland (1998)**: "Emergence is above all a product of coupled, context-dependent interactions. Technically these interactions, and the resulting system, are nonlinear: The behavior of the overall system cannot be obtained by summing the behaviors of its constituent parts." [23]
- **J. Goldstein (1999)**: "The arising of novel and coherent structures, patterns and properties during the process of self-organization in complex systems" [24]. Goldstein further elaborated to describe the common properties to identify system or computer simulations as emergent: (1) radical novelty; (2) coherence or correlation; (3) global or macro level; (4) dynamical; and (5) ostensive.
- **J. Fromm (2005)**: Fromm provides the following definition of emergent property: "A property of a system is emergent, if it is not a property of any fundamental element, and emergence is the appearance of emergent properties and structures on a higher level of organization or complexity." [1].
- **J. C. Mogul (2006)**: "Complex systems often behave in unexpected ways that are not easily predictable from the behaviors of components – emergent behavior. List of properties common to some or all instances of emergent misbehavior: Inherently hard-to predict behavior, Sudden changes in behavior, Amplification of seemingly minor behaviors" [18].
- **G. Marsh (2009)**: "The emergent properties of the collective whole do not in any transparent way derive from the underlying rules governing the interaction of the system's components" [15].
- **C. Szabo and Y.M. Teo (2013)**: "Complex systems often exhibit properties that are not easily predictable by analyzing the behavior of their individual, interacting components. These properties, called emergent properties, are increasingly becoming important as software systems grow in complexity, coupling and geographic distribution" [14].
- **O'Toole et al. (2014)**: "Emergence can only occur in systems composed of

autonomous parts, agents, who interact in dynamic, non-deterministic ways" [25].

Although the above definitions differ significantly, there exist some common characteristics covered by the different pairs of definitions. One of the tasks for us in this research was to study the different definitions of emergence and identify the various features that characterize emergence and then represent them in an ontology that includes a taxonomy of the features and definitions. In our literature review, we found many features of emergence which are used by authors to present their ideas. Figure 1 presents a concept lattice with emergence features (gray shaded rectangle) and referenced authors (empty rectangle). We used the technique of Formal Concept Analysis (FCA) [26] to develop this lattice to represent the mesh of definitions existing in the literature. The eight features included in the review are: Levels, Interactions, Radical Novelty, Unpredictability, Irreducibility, Dynamical, Coherence and Decentralized. These features appear in the definitions in [12, 27, 21, 22, 23, 24, 28, 29, 30, 1, 13, 19, 18, 17, 8, 15, 31, 32, 11, 25, 33]. Some of these definitions are presented above. Below we describe these features. Some of them are relatively simple to explain in intuitive way. But then some others are quite involved and even not well defined.

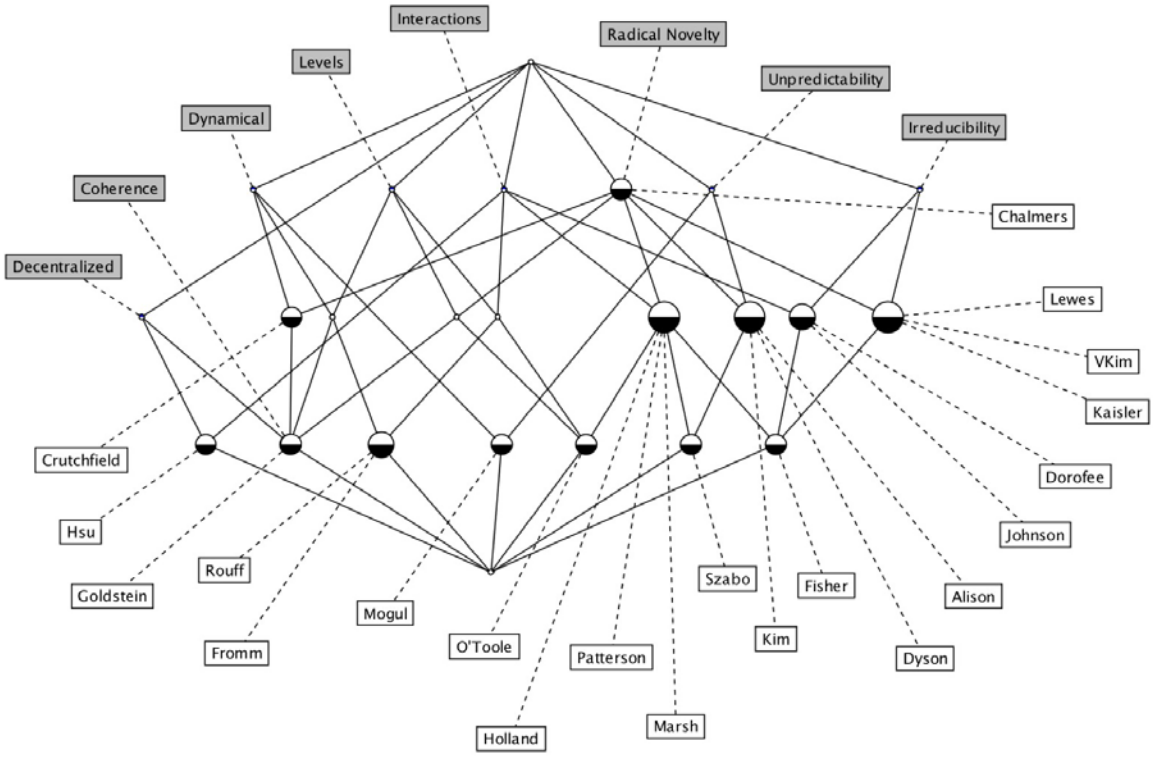
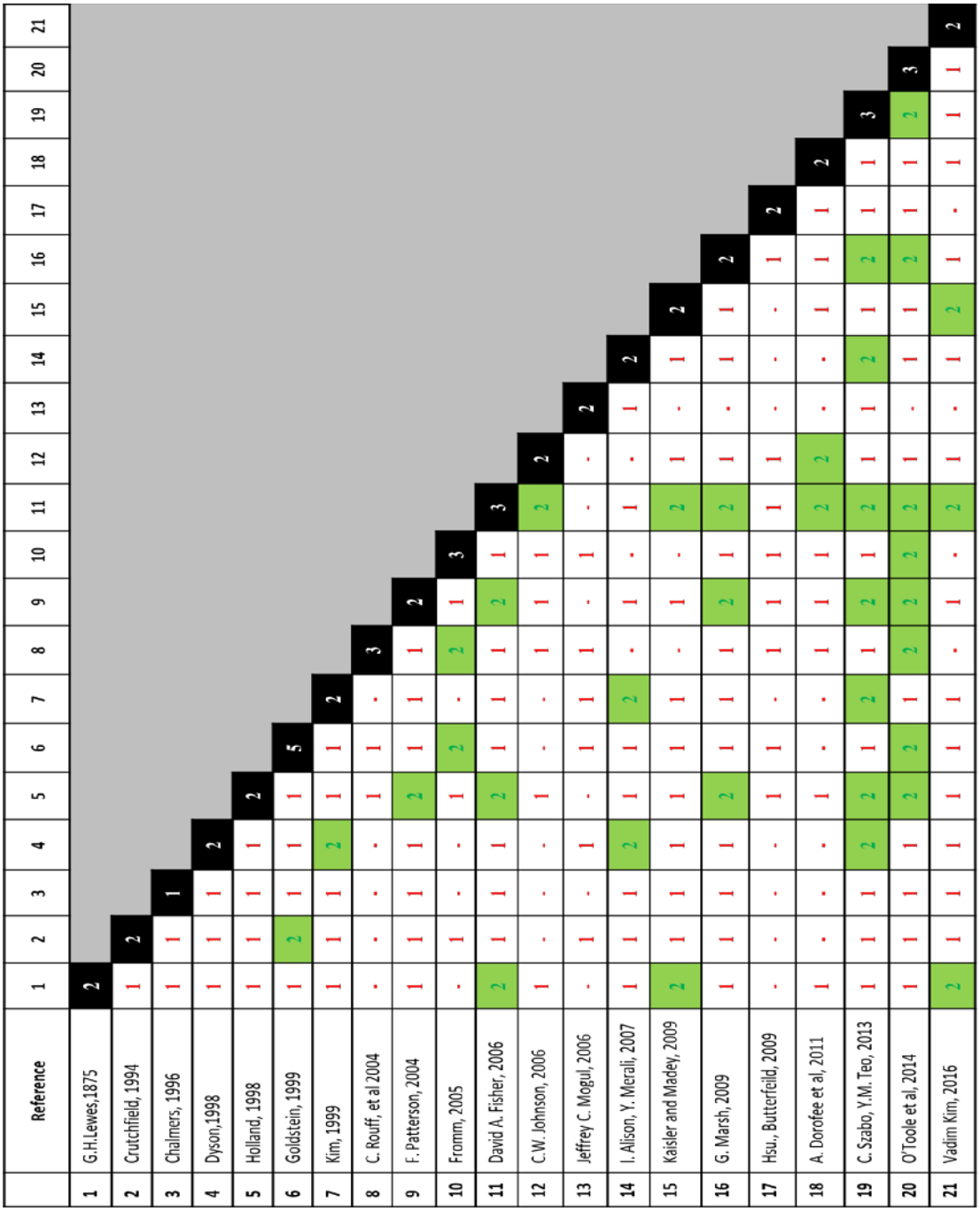


Figure 1. Concept Lattice with Emergence Features and Corresponding Reference using Formal Concept Analysis



	Reference	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	G.H.Lewes,1875	2																				
2	Crutchfield, 1994	1	2																			
3	Chalmers, 1996	1	1	1																		
4	Dyson,1998	1	1	1	2																	
5	Holland, 1998	1	1	1	1	2																
6	Goldstein, 1999	1	2	1	1	1	5															
7	Kim, 1999	1	1	1	2	1	1	2														
8	C. Rouff, et al 2004	-	-	-	-	1	1	-	3													
9	F. Patterson, 2004	1	1	1	1	2	1	1	1	2												
10	Fromm, 2005	-	1	-	-	1	2	-	2	1	3											
11	David A. Fisher, 2006	2	1	1	1	2	1	1	1	2	1	3										
12	C.W. Johnson, 2006	1	-	-	-	1	-	-	1	1	1	2	2									
13	Jeffrey C. Mogul, 2006	-	1	-	1	-	1	1	1	-	1	-	-	2								
14	I. Alison, Y. Merali, 2007	1	1	1	2	1	1	2	-	1	-	1	-	1	2							
15	Kaisler and Madey, 2009	2	1	1	1	1	1	1	-	1	-	2	1	-	1	2						
16	G. Marsh, 2009	1	1	1	1	2	1	1	1	2	1	2	1	-	1	1	2					
17	Hsu., Butterfeild, 2009	-	-	-	-	1	1	-	1	1	1	1	1	-	-	-	1	2				
18	A. Dorofee et al, 2011	1	-	-	-	1	-	-	1	1	1	2	2	-	-	1	1	1	2			
19	C. Szabo, Y.M. Teo, 2013	1	1	1	2	2	1	2	1	2	1	2	1	1	2	1	2	1	1	3		
20	O'Toole et al, 2014	1	1	1	1	2	2	1	2	2	2	2	1	-	1	2	2	1	1	2	3	
21	Vadim Kim, 2016	2	1	1	1	1	1	1	-	1	-	2	1	-	1	2	1	-	1	1	1	2

Figure 2. Overlap of Features between Definitions

- Levels: This feature refers to the idea that for emergence to exist, there must be at least two levels: (1) the level of a collection of agents, and (2) a higher level that is above the agents which interacts with the agents, but also has some attributes that are associated with the whole systems of agents, and not with any of the single agents. In general, there may be more than two levels.
- Interactions: This feature refers to an exchange of information, and possibly actions, between the agents and between the levels. Interactions may be passive, like taking measurements (observing) of the agents, or active - passing messages between agents and between levels.
- Radical Novelty: This feature refers to something that is "genuinely new", like new structures, patterns of behavior or properties.
- Unpredictability: It means that there is no prediction possible that is more efficient than simulation.
- Irreducibility: A behavior is irreducible if a property cannot be deduced from the properties of its constituent parts. This concept seems to be semantically equivalent to the concept of unpredictability. Since this is a very crucial concept to the understanding of emergence, we devote a whole section to it later in the report.
- Dynamical: This feature refers to the fact that interacting agents are (often non-linear) dynamical systems and thus emergence is the result of changes that take place over time.
- Coherence: Logical consistency or quantitative continuation (no erratic jumps).
- Decentralized: This feature refers to decentralized control, i.e., the agents are not controlled by a central controller.

We also present the overlap between these definitions. Figure 2 shows the number of features in each definition (number in black box), along with the overlaps, i.e., same number of features two definitions used (numbers in red, with more than two overlaps in green box). Based on the above definitions of features, it seems that at least three of the features can be combined into one: Irreducibility, Unpredictability and Radical Novelty. We will term this group simply "Irreducibility". In the following subsection, we discuss the notion of Irreducibility in more detail.

2.2 On Irreducibility in Mathematics

In this research, we have been seeking explanations to the concept of emergence with the objective of expressing emergence in formal terms, i.e., terms that are grounded in mathematics and logic. Mathematics provides a number of examples of irreducibility. We mention some of them below.

1. Irreducible element, e.g., a polynomial – not a product of other elements.
2. In representation theory, an irreducible representation is a nontrivial representation with no nontrivial proper subrepresentations.

    2.1. A representation makes an abstract algebraic object more concrete by describing its elements by matrices and the algebraic operations in terms of matrix addition and matrix multiplication.

    2.2. The algebraic objects to which representation theory applies can be viewed as particular kinds of categories, and the representations as functors from the object category to the category of vector spaces.

3. An irreducible fraction (or fraction in lowest terms) is a vulgar fraction in which the numerator and denominator are smaller than those in any other equivalent fraction.

4. In universal algebra, irreducible can refer to the inability to represent an algebraic structure as a composition of simpler structures using a product construction; for example, subdirectly irreducible.

5. A topological space is irreducible if it is not the union of two proper closed subsets. This notion is used in algebraic geometry, where spaces are equipped with the Zariski topology; it is not of much significance for Hausdorff spaces. See also irreducible component, algebraic variety.

6. There are more examples of irreducibility in mathematics, but the above ones look more like independent, while the rest seem to be just special cases of the above.

2.3 Computational Irreducibility

We are devoting a subsection to another idea of irreducibility that has been developed by Stephen Wolfram [34].

The concept of computational irreducibility (that some complex computations are not amenable to short-cuts and cannot be "reduced"), is ultimately the reason why computational models of nature must be considered in addition to traditional mathematical models. Wolfram terms the inability to shortcut a program (e.g., a system), or otherwise describe its behavior in a simple way, "computational irreducibility". The empirical fact is that the world of simple programs contains a great diversity of behavior, but, because of undecidability, it is impossible to predict what they will do before essentially running them. The idea demonstrates that there are occurrences where theory's predictions are effectively not possible. Wolfram states several phenomena are normally computationally irreducible.

The main aspect of this theory is that nature is a computation. This idea seems appropriate to discussing emergence. After all, when we analyze various phenomena, we always look at a progression, e.g., in time, of the phenomenon/system state. In our simulations of UAVs, we were representing agents as dynamical systems evolving over time. While the UAVs were changing their locations and other parameters, the value of the "next" parameter was actually computed. So there was a computational process behind all of

the assessments of the physical parameters.

Wolfram's idea of "computational irreducibility" is analogous to irreducibility in mathematics. He considers program instructions, executed one-by-one in a sequence. He claims that some of the programs consist of simple instructions, but still produce very complex behaviors. The issue is whether the state of computation can be predicted using a mathematical formula that could take in the initial state and one number, n, which is the number of computation steps, and would output the value of state after the n steps. In some cases, this is possible, but in many, perhaps most, not possible. Our understanding is that this is what Wolfram calls "computational irreducibility".

An example of a formula for predicting the state after n is a look-up table. Intuitively, it is related to automata that have repetitive pattern. And clearly related to the idea of using the metric of *variety* – since only some of the states are visited during the execution of a process.

Wolfram links computational irreducibility with *undecidability*. However, it is not clear what that link is precisely. In particular, it is not very clear how it is different from the notions of irreducibility in mathematics where irreducibility always implies some type *closure*, i.e., first a class (a structure) S is defined and then a proof is sought to show that a specific structure s is not within the class S. The question is whether an extension of the language is needed to capture the intuition behind emergence.

The notion of undecidability in the theory of computation is related to the notion of *incompleteness* in logic. Historically, first, Gödel proved two incompleteness theorems, which can be paraphrased as follows:

1. Any consistent formal system F within which a certain amount of elementary arithmetic can be carried out is incomplete; i.e., there are statements of the language of F within which a certain amount of elementary arithmetic can be carried out is incomplete; i.e., there are statements of F that can neither be proved nor disproved in F.

2. Assume F is a consistent formalized system which contains elementary arithmetic. Then while it is possible to express a statement $Cons(F)$ (meaning F is consistent), but $Cons(F)$ is not provable within F.

Then undecidability results were proven by Turing and Church for functions. Thus, undecidability and incompleteness are closely related.

However, undecidability still applies to the same formal system, i.e., while the decidability can be proved within a given formal system, it still can be decided in a stronger system. Perhaps this could give us an idea of going after to use it for emergence? We have not investigated this idea any further.

It is perhaps worthwhile to mention that Wolfram claims that computational irreducibility is not due to just the lack of computational power. Instead, the computation to make the prediction needs to be equivalent to the computation that the other computation is trying to predict. So, he introduces the principle of *equivalence* of computations. He claims that most natural processes are like computations, and even though they are simple, they can produce complex results. All such programs are then equivalent (meaning complex). Complex computations are irreducible. If one designs a computation to predict the behavior of a complex computation, such computations, in order to predict the outcome, perform

computations that are not more sophisticated (i.e., they are in the "complex" equivalence class) than the behaviors they are trying to predict; they simply perform the computation of the behavior. Moreover, Wolfram states that in the traditional analysis computational complexity and undecidability are related to a specific problem. But the programs that Wolfram investigates are not designed to solve any specific problem. So, the only issue with the analysis of this kind of computations is to predict the value of the computation at time *t*.

Unfortunately, Wolfram does not use mathematics to present his results. Instead, he uses simulation; in fact, most of it is simulation of cellular automata. This is not surprising, considering he is the guy behind the Mathematica simulation framework. So, in summary, while the idea of investigating emergence by considering computational processes as models of processes occurring in nature is appealing, Wolfram's ideas don't seem to be sufficiently formalized so far. This opinion has been expressed by many scientists in various publications, so our conclusion is not novel. The only point we are making is that we believe the idea of computational irreducibility is interesting and worthwhile pursuing in our future research.

2.4 Classification/Taxonomy of Emergent Behaviors

Based on our extensive literature review, there is no formal, universally agreed definition of emergence. However, emergence has been identified for many different types of processes. And a need for dealing with emergence has been documented and strongly supported. One of the directions of dealing with emergence is to identify the various features of emergence and develop classifications of this phenomenon based on these features. In this section, we describe some of those attempts.

A taxonomy of different types of emergence can be defined based on the feature of (discussed earlier in this report) and the relationships between the levels of the system (micro- and macro-levels). Understanding the distinction between these types is necessary to understand the context in which emergence can be detected. Such a taxonomy addresses the following objectives [10]: 1) to understand what emergent behavior is, 2) to understand the underlying principles, and 3) to identify emergent behaviors apart from other phenomena. Table 1 provides a list of some of the classification/taxonomy types of emergent behaviors found in the literature. The table also lists the criteria on which the different classifications are based.

One of the earliest emergent behavior taxonomies was presented by Chalmers [35] in 2002, which distinguishes emergence as weak and strong emergence. This classification is based on the feature of *deductibility*. *Strong emergence* is not deducible even in principle from the laws of the low-level domain, while *weak emergence* is only unexpected, given the properties and principles of the low-level domain.

Mark A. Bedau [36] extends Chalmers's taxonomy by adding the third type of emergence called *nominal*. Nominal emergence is the appearance of a macro property in a system that cannot be a micro property.

William Seager [37] discussed two kinds of emergence: benign and radical. Benign and radical emergence differ in their description of behavior. If one can find a descriptive or explanatory scheme describing the behavior of the system, the behavior is benign, otherwise it is radical.

Maier [38] gave four categories of emergence: *simple*, *weak*, *strong* and *spooky*. Simple emergent property can be readily and predictably produced in lower complexity, abstracted models of the system. Weak and strong emergence have the same meaning as described above. Spooky emergent property, on the other hand, is inconsistent with the known properties of the system's components. The spooky property is not reproduced in any model of the system, even one with the complexity equal to that of the system itself.

Mogul [18] provided a taxonomy of emergent mis-behaviors. The taxonomy presents a list of undesirable behaviors in software systems. Gore and Reynolds Jr. [39] presented an exploration based taxonomy for validating emergent behaviors rather than simulation. They presented taxonomy based on three orthogonal dimensions: reproducibility (repeatability of a simulation for a given set of inputs), predictability (increase the efficiency of exploration process) and temporality (distinguish between process of achieving a final state and residing in the final state). The authors claim that the taxonomy is robust, comprehensive and suitable to use with established emergent behavior exploration methods.

Yaneer Bar-Yam [40], Jochen Fromm [1] and O. Thomas Holland [10] distinguish five emergence types to which they assigned numbers instead of names. Bar-Yam presented a taxonomy consisting of four types of emergence based on particles and ensembles. A particle is a single acting agent or entity while an ensemble is a group of particles. In Type 0, no interactions occur between particles and behavior is seen strictly on the particle level. All the other types of behaviors involve ensembles. Type 1 behaviors are only unexpected given the properties and principles at the component level while Type 2 behaviors cannot be found in the properties of the system's lower level. Type 3 classifies the emergent behavior of systems that arise out of the interactions with the environment. Fromm [1] provided four types (I-IV) taxonomy based on feedback types and causal relationships between micro and macro level. We have used Fromm's approach in our ontology development. Detailed explanation is presented in the next section. Holland [10] provides an extension to Fromm's taxonomy. Holland also dispenses Fromm's concepts of "strong emergence" and "supervenience" and instead introduces the use of evolutionary agents at the local levels weherein feedback from either the global or local levels can cause individual entities to add or delete from their control rule set. Holland presented definitions of five types of emergent behaviors with four subtypes, with discriminating features as feedback types and pattern formation at the scale of observation.

Table 1. Different Classifications and their Criteria

Reference	Criteria	Classification/Taxonomy
Chalmers, 2002	Deductibility (Earliest EB taxonomy)	{Weak, Strong}
Bedau, 2002	Extension to Chalmers	{Weak, Strong, Nominal}
Bar-Yam, 2004	Resolution and Interactions	{Type 0, Type 1, Type 2, Type 3}
Fromm, 2005	Feedback and Interactions	{Type I, Type II, Type III, Type IV}
Seager, 2006	Description of behavior	{Benign, Radical}
Mogul, 2006	Undesired behaviors in software systems	Emergent Mis-behavior taxonomy
Holland, 2007	Feedback and Patterns	{Type 0, Type 1, Type 2, Type 3, Type 4}
Gore, 2007	Exploration based taxonomy - three orthogonal dimensions	{Reproducibility, Predictability, Temporality}
Maier, 2015	Complexity and System Classification	{Simple, Weak, Strong, Spooky}

2.4.1 Fromm's Taxonomy. As we stated earlier, since we selected the taxonomy provided by Fromm [1] to use for the development of a first cut of an ontology, which we use for the demonstration of the use of an ontology for detection of emergence, we provide a brief overview of this taxonomy below.

Fromm classified emergent behaviors into four different types based on interactions between micro and macro levels. Each type is further subdivided into two sub-types. Type I contains the simplest intentional emergent phenomena with a single feed-forward relation, e.g. clock, computer program, wave front. Type Ia is simple intentional emergence with no feedback, thus the behavior is deterministic and brittle. Type Ib, on the other hand, is simple unintentional emergence with no feedback but with peer-to-peer interaction.

From an engineering perspective, a particularly interesting case is type II emergence which encompasses systems exhibiting self-organization and other useful properties. Type II is weak emergence with top-down feedback. Type IIa includes negative feedback which imposes constraints on actions of the agents. The examples include flocking and ant foraging, whereas Type IIb refers to undesirable emergence with positive feedback. Examples include crashes and bubbles in the stock market. Type III and IV have the highest level of complexity. Type III phenomena are characterized by multiple feedback loops, both positive and negative. This type is common in open systems with high complexity and it is usually associated with activator-inhibitor systems, e.g. patterns in biological entities, prisoners dilemma, as well

as evolutionary and adaptive systems. Type IIIa is a combination of types IIa and IIb with short-term positive and long-term negative feedback. Type IIIb consists of multiple feedback loops and many constraint generating processes. Type IV contains the emergence of completely new complex systems (e.g. culture, life). We focus on first three types of emergence. Type IV is out of scope for this project and our study. The classification is summarized in Figure 3.

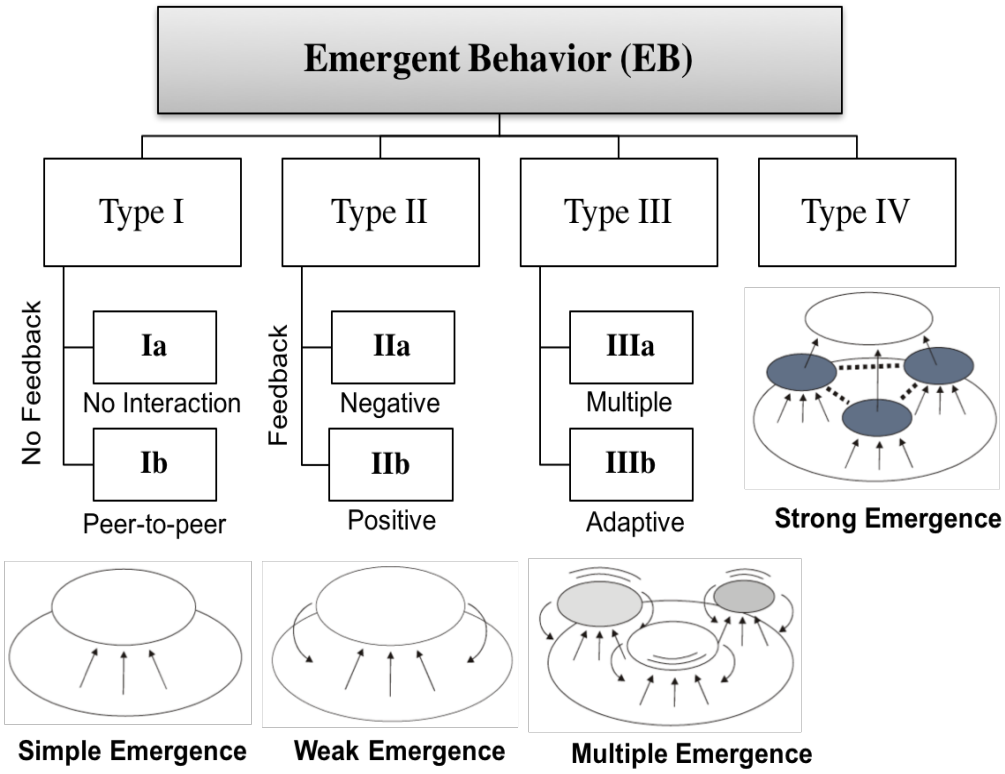


Figure 3. Taxonomy of Emergent Behaviors by Fromm [1]

2.5 Emergence in Swarms of UAVs

Many studies of undesirable emergent behavior are not directly relevant to the domain of UAVs for ISR (Intelligence, Surveillance, and Reconnaissance). Some studies (e.g., [41]) have relevant models such as flocking behavior but have limited coverage on the domain of UAVs. Nevertheless, they are beneficial from the point of view of the formalization of the concept of emergence, as well as sources of examples of emergent behaviors and behavior control rules. In particular, the simulation studies like in [3] have proved that it is possible to observe emergent behaviors similar to the ones occurring in nature by using distributed simulation in which each object is simulated based on a set of behavioral rules.

In [42], simulations are used to investigate the behaviors of swarms of UAVs. Two distinct scenarios were considered: mapping a contaminant plume and detecting and tracking vessels crossing a body of water. UAV swarming behaviors were explored and how they could be controlled by ground pilots (centralized control) to accomplish both of these scenarios. These scenarios implement rules from Reynold's BOIDS [3] which demonstrates the emergent

behavior of flocking in UAVs.

In our research work, as well, we consider scenarios in which multi-agent (UAV) systems are designed to carry out missions such as tracking or surveillance of targeted search areas/facilities. It is a known fact that these kinds of systems, i.e., systems with autonomous agents, exhibit various types of behaviors, some of which may be highly undesirable with respect to the mission. Potential forms of undesirable emergent behavior are:

- ***Thrashing***: UAVs keep moving back and forth between facilities due to multiple requests.
- ***Poorly covered or non-covered facilities:*** Some facilities or part of search areas get ignored and activities that should be detected are missed.
- ***Saturation:*** One facility gets too much attention at the expense of other facilities.
- ***Collision:*** Two moving agents executing their control policies that fail to account for the dynamics of other objects operating in the same space, which results in a collision.
- ***Imbalanced use of resources:*** E.g. all agents are tracking the same area.

In the work documented in this report, we implemented and detected such types of undesirable emergent behaviors in a multi-UAV system. Next sections present the methods and approaches we applied for the analysis and detection of emergent behaviors.

3 METHODS, ASSUMPTIONS AND PROCEDURE

3.1 Targeting Emergence: EBS (Emergent Behavior System) Frame- work

3.1.1 Emergence in Levels of the System. Many authors, cf. [1, 10, 43, 44], agree that the notion of emergence involves the existence of levels in the system. Emergence can be summarized as a characteristic of system where properties and behaviors appear at the system (macro) level that were not explicitly implemented. These properties arise dynamically from the interactions between entities at the component (micro) level, and cannot be reduced to the properties or behavior of the individual entities [25]. This relationship between micro and macro level is presented in Figure 4. Interactions of components at the micro level result into group formation (as in flocks of birds) at the macro level.

The system, the behavior of which is a result of emergent phenomena, is called Emergent Behavior System (EBS) [45]. We will also use this term to refer to systems that exhibit emergent behaviors including the multi-UAV systems. Popular examples of types of systems that exhibit emergent behavior include flocking system, particle system, stigmergy system and traffic system. To better understand these kinds of system and their exhibited behaviors, models are needed to explore their properties. The simulation technology exists today to build such emergent behavior systems, but the theoretical framework to support modeling and analysis is still in its infancy. We need an EBS framework to establish a foundation and context for simulation development and analysis of emergent behaviors. The next sub-

section discusses in detail the proposed EBS simulation framework presented by us at the SpringSim 2017 conference [46]. But before going into details of the framework, it is essential to understand the constituent elements of any EBS which are needed to be identified or implemented first to study the behavior of the overall system. The EBS comprises of the following elements:

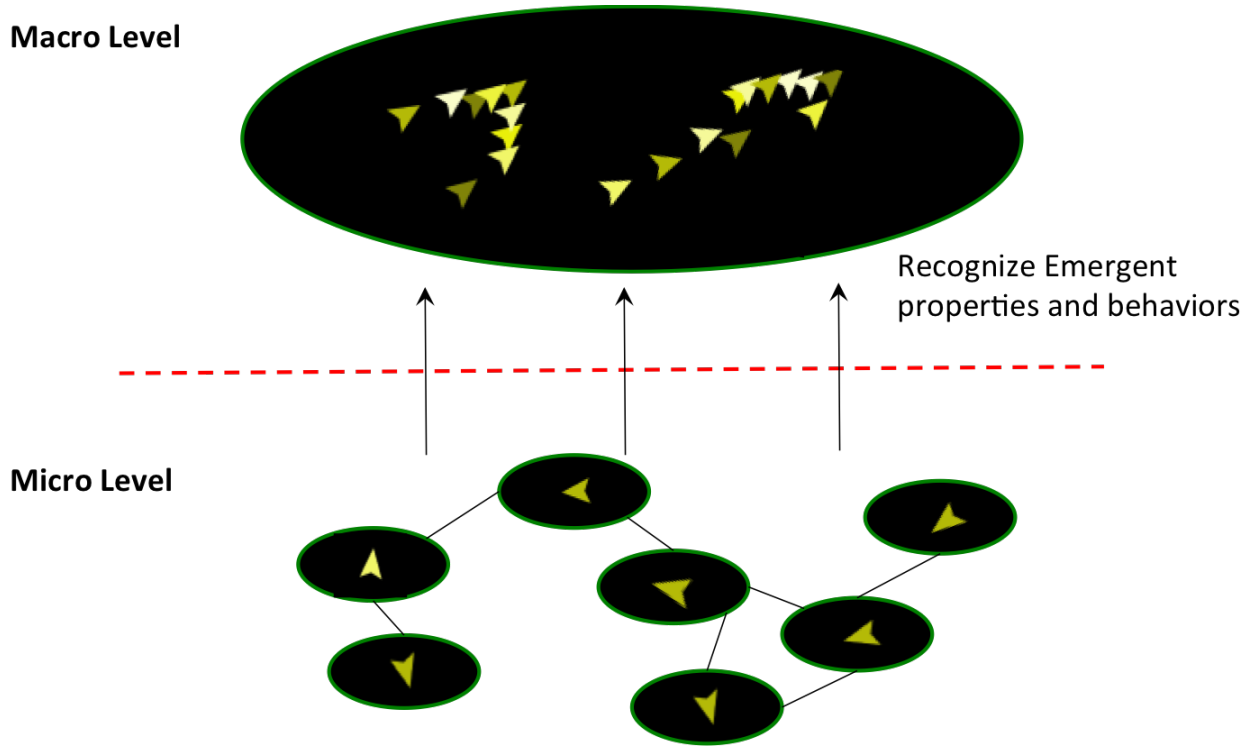


Figure 4. Relationship between Micro and Macro Levels

1. *Agent:* An agent is any physical or logical entity/element of a system that is designed/specified through an engineering process. The distinguishing characteristic of agent is some amount of autonomy, i.e., the capability of making own decisions in order to accomplish a specific goal.

2. *Interaction between Agents*: An interaction represents the relationship between agents and defines how agents interact with other agents and environment. In particular, agents can interact via exchange of messages, or by observing characteristics of other agents, either directly, or by reading traces that agents leave in their environment.

3. *Agent Environment:* An environment corresponds to a set of conditions in which an agent must perform its functions. It may include influences on the agent, like forces, or any physical inputs.

4. *System*: A system is composed of agents, environment and their interactions.

5. *External Commander*: An external entity that observes the behavior of the system

and possibly controls it via some control means (control inputs).

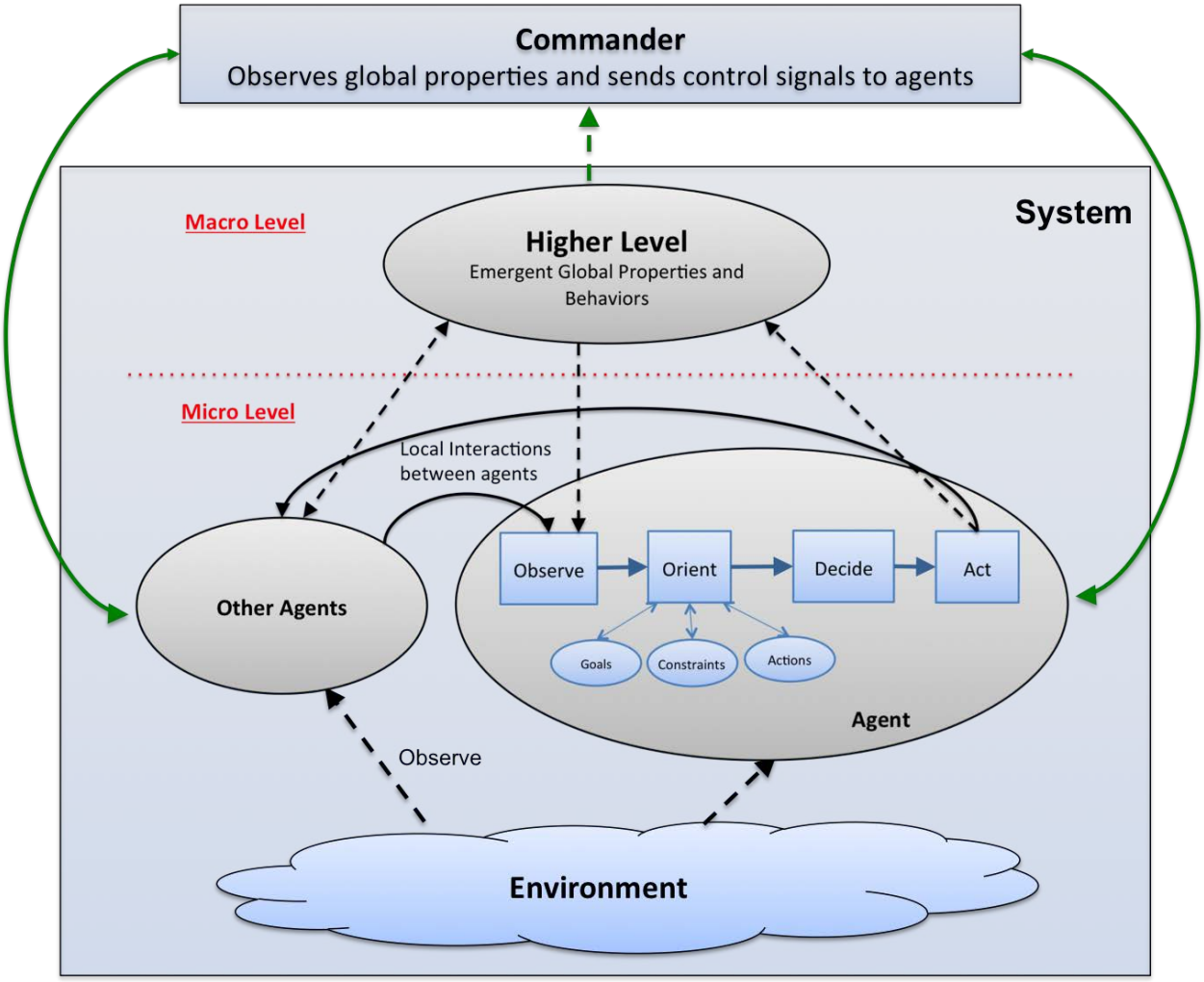


Figure 5. Agent(s) in EBS.

3.1.1.1 Agent. In literature, although there is no widely agreed definition for an agent, there is a general consensus that autonomy is central to the notion of agency [47]. Among various definitions, we consider the general definition of an autonomous agent as given in [48]: "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future". Agent, in Figure 5, represents an independent entity that interacts with other agents and environment in the system. Each agent has a set of attributes that describe the state of the agent and a set of specified policies that define how an agent behaves with respect to changes in its environment.

3.1.1.2 Interaction between Agents. To model the functioning of an agent, we con- sider that each agent executes the OODA loop, as shown in Figure 5. OODA is the acronym for **O**bserve, **O**rient, **D**ecide, **A**ct [49]. The OODA loop is appropriate for modeling autonomous

agents that operate and interact with other agents through a rational decision-making process that follows a cycle of observation, orientation, decision-making, and action.

Multiple agents in the system can all individually run their OODA loops along with interaction with each other. For instance, the Action of an agent can be observed by one or more other agents and can, in turn, influence the decision of the other agent (shown as solid black arrows in Fig. 5). Furthermore, in our model we also define the following three datasets associated with the agent's Orient function:

- Goal Space: defines the goal of the agent
- Action (Policy) Space: defines the set of actions available to the agent to achieve its goal
- Constraint Space: set of constraints (restrictions) on actions

The sections that follow explain these spaces in the context of our multi-UAV system (persistent surveillance) scenario.

3.1.1.3 Agent Environment. The environment is observable medium with which agents interact and is not considered as being an agent in its own right; that is, it is passive and global (it does not actively assert behavior and it potentially affects all agents) [50, 51]. The environment is simply used to provide information to agents such as the spatial location of other agents in its vicinity or air conditions etc., depending on the domain or objective of the system.

3.1.1.4 System. A system comprises agents, their interactions, and environment. A system has a clearly defined boundary with well-defined inputs and outputs (if appropriate) [50]. As mentioned above and in our framework, as well, we model the behavior of a system by two levels, i.e. macro and micro levels. The taxonomy of different types of emergent behaviors is based on the relationship between these macro and micro levels:

- *Micro Level* consists of entities/agents that interact with each other by observing the environment around it (local information) and/or directly by either exchanging messages or observing the results of the actions (outputs) of the other agents.
- *Macro Level* is a virtual level where system level properties (global) are generated and maintained.

There is a bi-directional link between the two levels in the system and in particular the downward causation (black dashed lines) that pass influences from the macro to the micro level. Causation means that an emergent phenomenon that exists at the macro level impacts the agents at the micro-level by constraining their behavior, or their environment in some manner. At the same time, the upward link connects the low-level agents with the higher level.

3.1.1.5 External Entity - Commander (Observer). Ronald et al [52] state that without an observer, the issue of emergence could not arise at all. Also, the authors in [53]

state in their paper, "…. the emergent aspect of a phenomena is related to the point of view of an observer of this phenomena: it is not intrinsic to the phenomena, but related to the global system (phenomenon + observer)". This insight presents an important consideration for the EBS modeler, specifically for the applications where one wants to detect and control the emergent behaviors. Thus, when observing the phenomena in a natural or simulated system, the discovery of patterns and emergent features implies the existence of an observer. In our framework, *Commander* represents an external entity (observer) that observes the global properties of the system and takes necessary actions (i.e., sends control parameters to agents if necessary).

3.1.2 Emergence in Ontology. Since different emergence cases have different features, a generic representation formalism is needed in which models of all such behaviors can be specified. In other words, the representation formalism must be able to capture all possible cases of emergence. A language in which the features of each of the cases of emergence are represented in the same way is needed. Ontology has been widely used in agent-based modelling and simulation [54, 55, 56]. However, none of the works that we are aware of has demonstrated the capability of representing and classifying all possible cases of emergence in a system. In this project, we used the ontology to model the emergent behavior. The Web Ontology Language (OWL) [57] has been used as the formal language. Ontology of emergence is discussed later in the report.

3.1.2.1 Ontology Basics. In information sciences, "an ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse" [58]. An ontology can be represented in a formal language, e.g., OWL; it is then called a formal ontology. In general, ontology has four basic primitives: class, relation, individual and axiom:

- ***Class:*** also called Concepts or Types, represents a group of different Individuals, that share common characteristics, which may be more or less specific.
- ***Relation:*** ways in which classes and individuals can be related to one another.
- ***Individual:*** also known as Instances, are the base unit of an ontology; they are the things that the ontology describes or potentially could describe.
- ***Axiom:*** assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application.

3.1.2.2 Features of Emergent Behaviors. In order to develop an ontology to model emergent behaviors, we needed to investigate the features of emergent behaviors first. Table 2 lists the features of emergent behaviors that are described in the literature. We focused on three aspects of features of behavior modelling to facilitate detecting emergent behaviors:

Table 2. Features of Emergent Behaviors

Paper	Emergence phenomena	Features to characterize the emergence	Map to the ontology
Fromm, 2005 [4]	thermodynamics	pressure, volume, temperature	Class: Pressure, Volume,
	networks	path lengths, clustering coefficients	Class: Length
	bird flocks	direction, velocity, center of flock	Class: Direction, Velocity, Position
	world wide web	standards and constraints of the W3C, HTML, HTTP	
	free markets	supply, demand	
	economic infl	wage, product price	
	stripes and spots in animals coat	color	Class: Color
O'Toole et al, 2014 [4]	flocking	heading, position, the position of the centre-of-mass	Class: Direction, Position
	segregation	color, the number of agents in neighborhood with the same color	Class: Color, Direction
	ant colony	path, heading	Class: Direction
	game of life	alive, dead	
	gas particles	direction, velocity, color	Class: Direction, Velocity, Color
Miner et al, 2008 [6]	boids	density (area covered by agents divided by the number of agents)	Class: Density
Holland, 2007 [3]	gases	pressure, volume, temperature	Class: Pressure, Volume,
	flocking, schooling	distance	Class: Distance
	foraging ants	trails	
	free-market economies	consumer-production balance	
Parrish et al, 2002 [7]	fish schools	population size, velocity, dependence, direction, neighbor scaling	Classes: Velocity, Direction
Chan, 2011 [2]	game of life		
	boids	distance	Class: Distance
	Brownian motion		

- Supports both the modeling of behavior at the system-level and at the component-level.
- Distinguish internal (component) behaviors and external (between-component)

behaviors.

- The behavior of the system is the sum of the components' behaviors and their interactions.

3.1.3 EBS Simulation Framework. In this section, we present our approach [46] to developing a simulation system, where the fundamental idea is to explore emergent behaviors in EBS, i.e. analyzing the causal relationships between micro and macro levels of the system. The ultimate goal is to create a taxonomy consisting of types of emergence and providing mechanisms to control any kind of undesirable behavior exhibited by such systems, within this taxonomy. Figure 6 presents our proposed framework consisting of two components, i.e. *Conceptual Modeling* and *Experimentation*, in which the EBS is simulated. These are further divided into sub-components which are discussed in the subsections that follow.

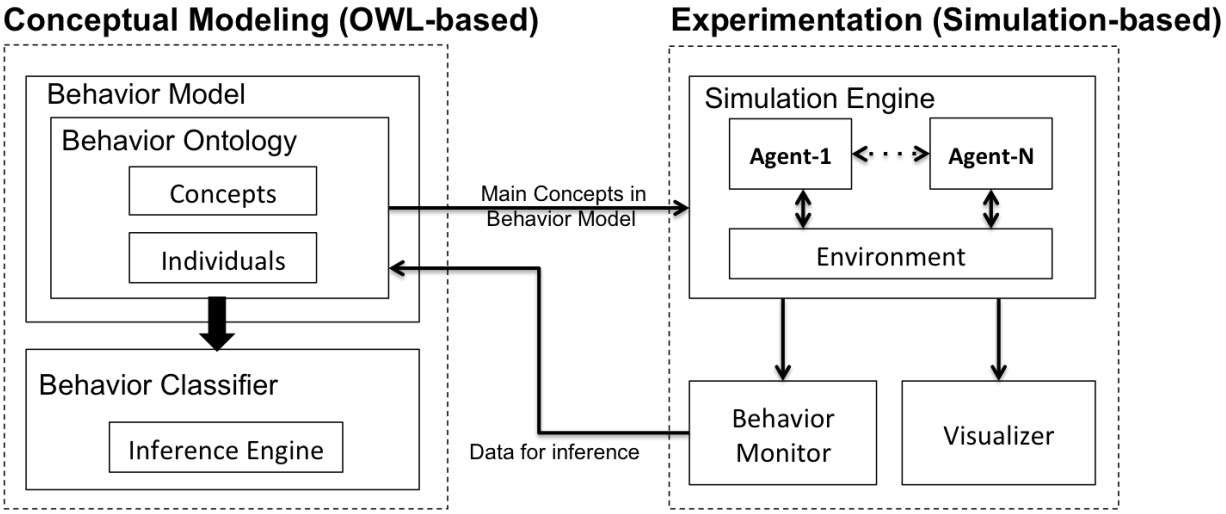


Figure 6. Proposed EBS Framework Showing OWL and Simulation Based Components

3.1.3.1 Behavior Model. In our EBS framework, system behaviors are modeled using an ontology. It is based on Nuvio (Northeastern and VIStology) ontology, a new foundational ontology developed by Northeastern and VIStology, Inc. The behavior ontology provides a common vocabulary for defining the concepts and relationships between those concepts for specifying behaviors of multi-agent systems. It aims to provide a formal approach to reasoning and classifying emergent behaviors. In this ontology, the behavior of multi-agent systems is represented by a Finite State Machine (FSM) diagram. Figure 7 shows how the concepts of behavior model are linked to Nuvio. Figure 8 shows the structure of the FSM representation in the ontology.

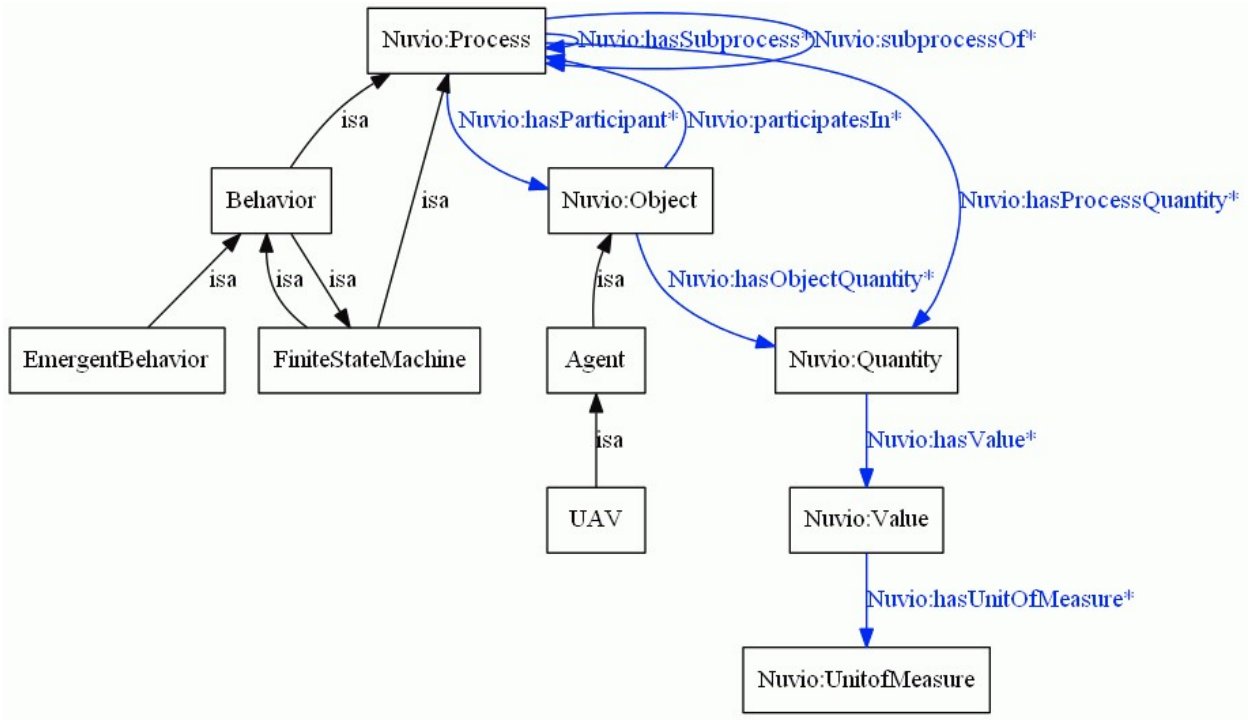


Figure 7. Behavior Modeling Concepts Based on Nuvio

Figure 9 shows the main ontological concepts for representing the structure of a multi-agent system. In a multi-agent system, a component is any physical or logical entity/element of a system that is designed/specified through an engineering process. A system is a set of components that interact with each other. The system consists of at least one component. A system could also be a component in a system of systems. A system could have several system attributes, such as input, output, interaction, goal, reference, and feedback. These attributes can be used to characterize the system behaviors.

We considered two types of behavior in a system:

- Component behaviors: this is a specification of how the dynamic state of a Component is allowed to evolve in time.
- System (group) behaviors: it is the aggregation of component behaviors and interactions between components.

The main concepts in the behavior ontology are mapped to the corresponding objects and functions in the agent-based simulation language.

3.1.3.2 Behavior Classifier. The main advantage of formal ontologies is that they can be used by software agents for automated inference, i.e., inferring facts that are only implicitly stated. The inference is carried out by inference engines (or reasoners). Reasoners can be incorporated into other algorithms. In our EBS, we use OWL reasoners to derive classifications of behaviors.

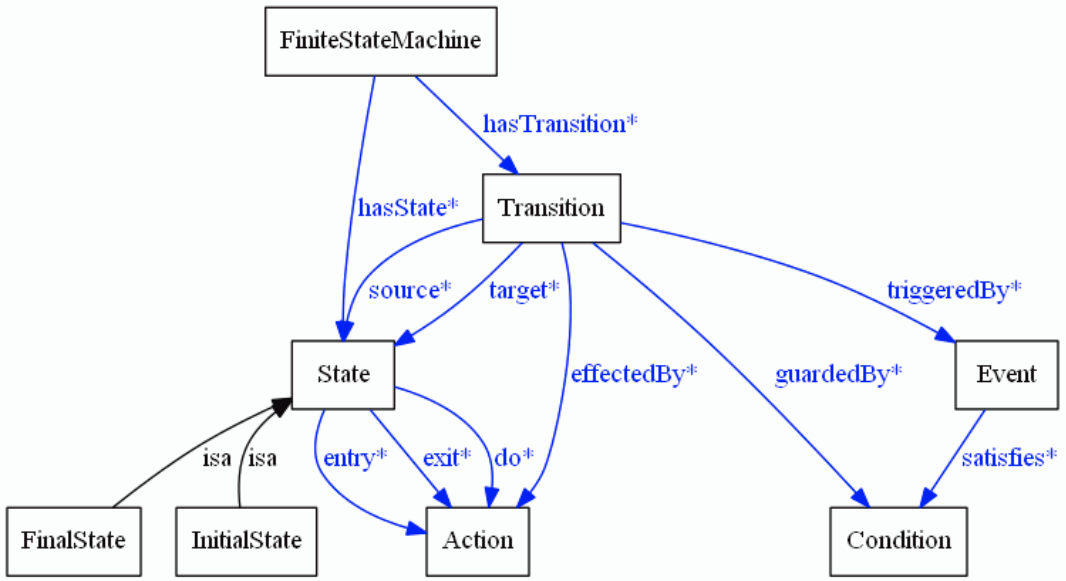


Figure 8. Ontology Concepts for Representing Finite State Machine

Figure 10 shows part of the class hierarchy of the behavior ontology. The boxes in this figure represent *classes*, i.e., collections of objects of the class type. In this ontology, we distinguish Behavior and Behavior Model. Behaviors are the actual behaviors that are described by some basic attributes, e.g., velocity, heading. Behavior models are the abstractions of actual behaviors that are described by other variables. For example, we use the feedback to define different emergent behavior types. The main behavior model class is *Emergent- Behavior*. Definitions of classes are given by the *axioms* expressed in OWL, which are used to define different types of emergent behavior. For instance, Equation 1 shows an axiom (using description logic) of the definition of Type IIa. Once the data of system behavior features collected from the simulation are imported to the ontology, OWL reasoners may be able to infer the behavior type automatically.

$$TypeIIa \equiv EmergentBehavior \cap \exists hasParticipant.(System \cap \exists sendNegativeFeedbackTo.Component) \quad (1)$$

3.1.3.3 Simulation Engine. The EBS simulation system utilizes an agent-based simulation engine to model and simulate the EBS. With a given abstract scenario specification (FSM) of a complex system under study as input, the model of an agent and system is designed and simulated using an available simulation platform. We have a variety of options that exist in literature [59]. For the purpose of this work, NetLogo [60] was selected as the platform for experimentation as it allows modeling and animation of an agent-like entities in a simulation environment supported by a scripting language, visual animator, and data output mechanisms.

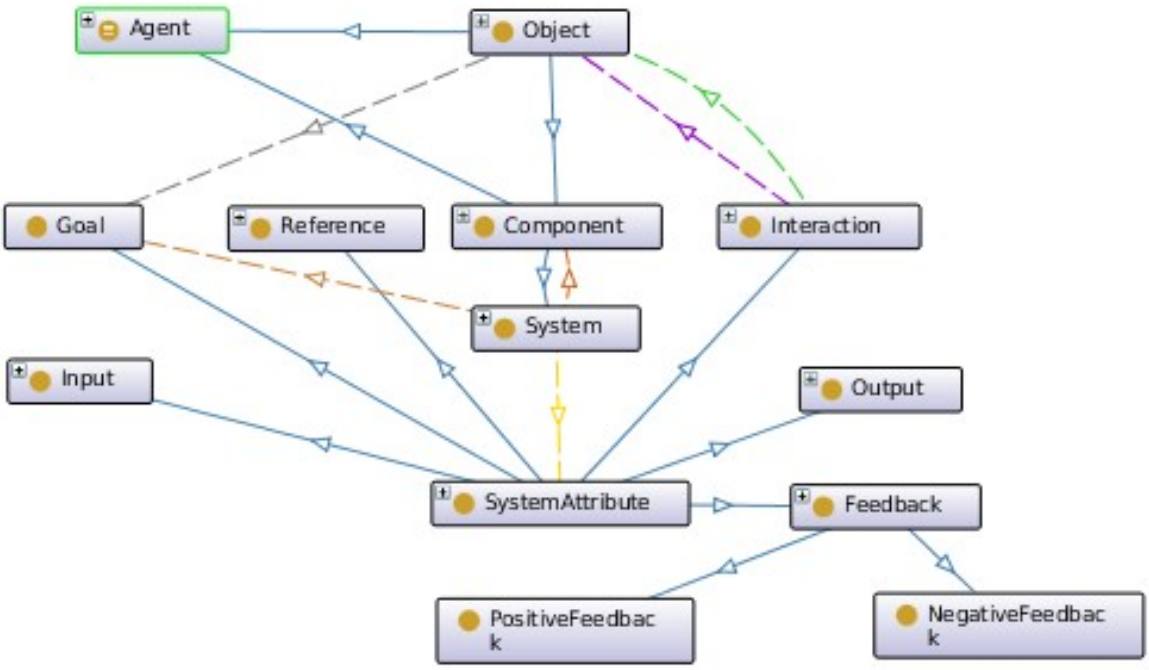


Figure 9. Main Ontological Concepts for Representing Structure of a Multi-Agent System

3.1.3.4 Visualizer. Visualizer provides a visual representation of the simulation. It may be used by the user to detect emergent properties arising at the macro level. NetLogo has a built-in visual animator window that we used to visually confirm the existence of emergence during simulation.

3.1.3.5 Behavior Monitor. Although Visualizer provides means to recognize emergence by visual inspection, we still need some kind of mathematical, or computer-supported, proof. This function is performed by *Behavior Monitor*. This module detects emergent behaviors by analyzing the global properties generated at the macro level of the system (using the data generated by simulation). In literature, many techniques exist to detect emergence, which range from statistical analysis to formal approaches. For the purpose of our current research, variable-based emergence detection [9, 10, 25] seemed to be the most appropriate choice. We discuss a metric called *variety* in the next section. Besides, Behavior Monitor is also responsible for sending some of the variables from simulation to the OWL model so that inference can be performed on the types of observed behaviors (type of emergence or good/bad behavior).

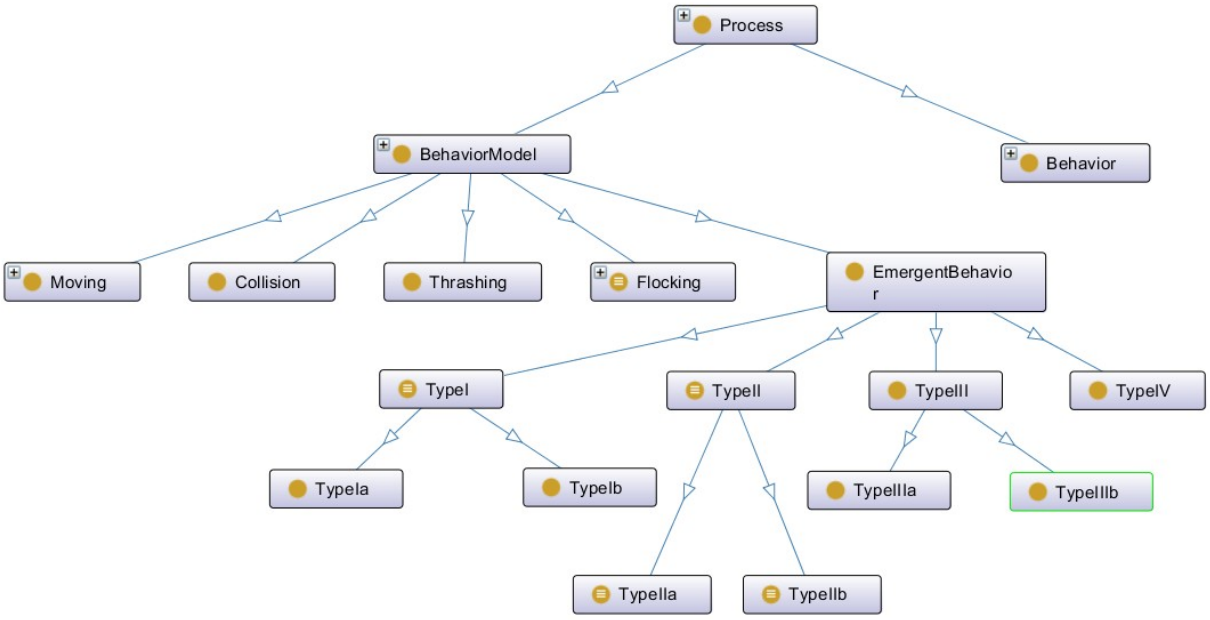


Figure 10. Classification of Emergent Behaviors

3.2 Scenario Formalization: Undesirable Emergent Behaviors in Swarms of UAVs

This section presents the formalization of the scenario that we selected to study in the current work. We address the persistent monitoring problem. The goal is to control the motion of UAVs to maximize the coverage and minimize the metric of *information age* (definition discussed in the next section) over the targeted area in the environment. The targeted area, termed as Plume, is a high priority enclosed area inside the environment, to be monitored by a UAV. The optimal control framework for each UAV is designed in such a way as to provide maximal coverage over the plume while minimizing the age of information about the plume in each location inside it.

In the coverage control, it is common to model knowledge of the environment as a non-negative density function defined over the mission space and usually assumed to be fixed over time. Some researches in persistent monitoring tasks involve dynamically changing the environment, but for simplicity we assume environment to be non-changing and stable. I.e., we do not vary the status of the plume in our simulations, although the monitoring of the plume does not depend on this assumption.

3.2.1 Problem Specification. A UAV, U, operates in a two-dimensional environment, $E \subset X \times Y$, where $X = Y = N$ are natural numbers that enumerate the cells of the environment. We consider a plume, $P \subseteq E$ as the targeted search area to which U provides persistent surveillance by measuring the environment at each time instant $t \in T = N$. Each measurement covers a subset of the environment $S(t) \subset E$. Assuming mission starts at $t_s \in T$, initial time, and ends at $t_f \in T$, the final time, the measurements are a sequence $S(t_s)$,

$S(t_s + 1), \ldots, S(t_f)$. Such sequences of measurements can be considered as paths that the UAV travels. Figure 11 shows the surveillance area demonstrating cells of environment and plume.

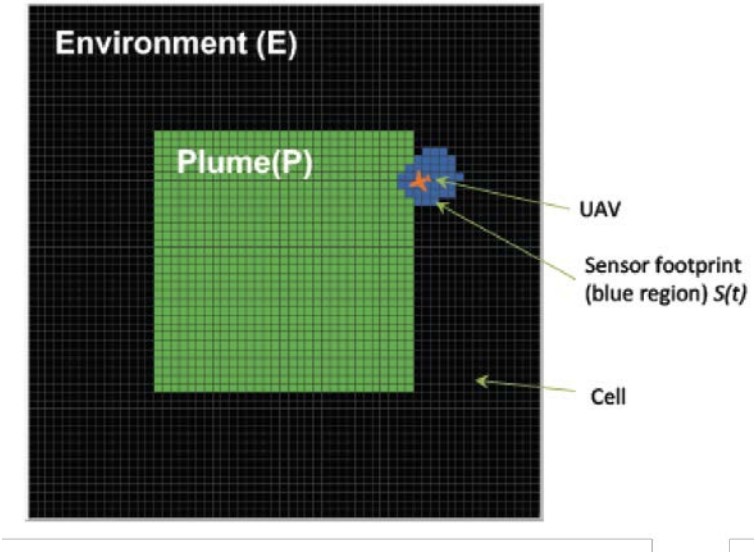


Figure 11. Surveillance Area (NetLogo Snapshot)

Here we are interested in monitoring the plume in some optimal way. In order to formulate the optimization problem, we need to define the objective function, the variables that can be manipulated in order to minimize the objective function, and possibly some constraints. Loosely speaking, the objective is to cover the plume "as best as possible". This needs to be expressed formally. Towards this aim, we use the notion of *information age*. At time $t > 0$, we will assign the age of zero to the cells of the plume $((i, j) \in P)$ that are measured at t, and then increase its age proportionally to the time. The rest of the cells of the environment E that do not belong to P are assigned the value of -1 in order to distinguish them from the plume cells:

$$A_{i,j}(t) = \begin{cases} 0 & (i,j) \in S \cap P \\ A_{i,j}(t-1) + 1 & (i,j) \in P \setminus S \\ -1 & (i,j) \in E \setminus P \end{cases} \tag{2}$$

The next step is to define a single measure of *coverage* of the plume. For this we just sum the ages of the cells in the plume computed by Equation 2.

$$I_{age}(P,t) = \sum_{i,j} A_{ij}(t), \quad (i,j) \in P \tag{3}$$

Note that Equation 3 represents an *instantaneous* coverage of the plume, P, i.e., it is the value of coverage at time t. However, since we are interested in the persistent surveillance of the plume, we need to define the problem of optimizing the mission, i.e., find a path of

the UAV that measures the plume so that the overall mission is optimal. The control decision then moves the UAV to a new position according to the dynamics of the system, i.e., the physical constraints due to the capabilities of the vehicle. Since this is a dynamical system, the response to control actions depends on the current state of the UAV system. We model the control action by the acceleration, *a(t)*, applied to the UAV at the particular time instance, which results in the new position of the UAV in the next time instance,

$$S(t+1) = f(S(t), a(t)) \tag{4}$$

$$a() \in [a_{min}, a_{max}] \tag{5}$$

The function, f, in Equation 4 represents the control law. Now let us assume that the UAV can make control decisions at each time instant, i.e., use a different control law at each time step. In that case, we can represent the control actions of the UAV as a sequence $\bar{a}$ = *(a(t$_s$), . . ., a(t$_f$))*.

How do we measure the quality of a mission (or path)? Here we propose to use the sum of the ages of the plume cells during the whole mission.

$$I_{age}(P, \bar{a}) = \sum_{t=t_s}^{t_f} I_{age}(P, t) \tag{6}$$

Note that $\bar{a}$ does not appear in Equation 6. However, the relation does exist via Equations 2 and 4.

Now we can formulate the optimization problem. The goal is to find a sequence $\bar{a}$ from all possible sequences a_k such that it minimizes the information age as defined by Equation 6. We define $S(t)$ as the area covered by U at t such that $S(t) \subset E$. Thus, for a given initial position, velocity and heading of the UAV following the dynamics in (1), find S_k for UAV such that overall age of the plume is minimum. Mathematically it is represented as:

$$\begin{aligned} &\underset{S(t)}{\text{minimize}} \quad I_{age}(t) = \sum_{t=t_s}^{t_f} \sum_{i,j} \sum_{z=1}^{N} w_{ij}(1 - \chi_{S_k(t)})(A_{ij}(t-1) + 1) \\ &\text{where} \qquad S_k(t) = f(S_k(t-1), a_k(t-1)), a_k \in [a_{min}, a_{max}] \end{aligned} \tag{7}$$

$$w_{ij} = \begin{cases} 1 & (i,j) \in P \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

3.2.2 Agent (UAV) Dynamics. Considering each UAV as a dynamic system (moving in 2D), the dynamic system is defined by the following components:

- x, y: location (state variables)
- v_x, v_y: velocity (state variables)
- a_x, a_y: acceleration (input variables)
- T: time interval between the time instant $k-1$ and k

The state transition function, $f = (v_x, v_y, x, y)$ is defined below:

$$\begin{aligned} v_x(k) &= v_x(k-1) + a_x(k-1) \cdot T \\ v_y(k) &= v_y(k-1) + a_y(k-1) \cdot T \\ x(k) &= x(k-1) + v_x(k-1) \cdot T \\ y(k) &= y(k-1) + v_y(k-1) \cdot T \end{aligned} \tag{9}$$

The above equations can be alternatively represented in matrix version. The state vector, q and input vector, u are given by:

$$\mathbf{q}(k) = [x(k), y(k), v_x(k), v_y(k)]^T \tag{10}$$

$$\mathbf{u}(k) = [a_x(k), a_y(k)]^T \tag{11}$$

The state transition function, $\boldsymbol{f}$, is given by the following equation:

$$\begin{aligned} \mathbf{q}(k) &= f(\mathbf{q}(k-1), \mathbf{u}(k-1), k) \\ &= \mathbf{A}.\mathbf{q}(k-1) + \mathbf{B}.\mathbf{u}(k-1) \end{aligned} \tag{12}$$

where **A** and **B** are matrices defined below:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ T & 0 \\ 0 & T \end{bmatrix} \tag{13}$$

3.2.3 Dimensionality Problem. We consider each of the entities i.e., single UAVs, as dynamical systems moving along a trajectory through the space defined by the Cartesian product of the system's parameters (inputs X, states Q and outputs Y, each being a vector). While all the systems are embedded in the same space and time, some of them additionally are coupled via their input/output parameters (e.g., exchange of messages either informing of their own locations or plans or requesting special movements by the peers). The conjoined space of all the participating systems can be modeled by the Cartesian product of all the system spaces $\Pi\ (X_i \times Q_i \times Y_i)$, usually denoted as $\Re^n$. The dimensionality of this space is relatively high and thus the analysis, detection, and control in such spaces are very complex. The exploration and analysis of such systems is further complicated by the fact that each of the parameters depends on time, and additionally, that each of the systems has its own dynamics, captured by the fact that the next state depends on the previous state and input, where next depends on the time parameter T_i representing the time interval between the state transitions. Since the inputs, states, and outputs for each of the systems are multidimensional, the dimensionality of the whole space is very high. The goal of this research is to reduce the dimensionality of the space in a principled way to make it practical to detect undesirable emergent behaviors. In order to do so, we use qualitative methods to study the behaviors exhibited by EBS. The use of such methods will lower the computational complexity of the algorithms for detecting undesirable behaviors with respect to a more traditional approach based on just a general dynamical system. We are using dimensional analysis and

Quantitative-Qualitative (Q^2) abstraction approach in the current work. These are discussed in detail in the following sections.

3.3 Qualitatively Different Behaviors

The approach for generating the qualitatively diff t behaviors in a system relies on the theory of similitude (or physical similarity). This theory has been used extensively in physics and engineering, in particular to model behaviors of phenomena that occur due to the interactions of particles, e.g., in heat and mass exchange. For instance, Osborne Reynolds, in 1895, defined a criterion, called *Reynolds number (Re)*, to describe transition between the two kinds of flow (turbulent and laminar). In similarity theory, numbers like *Re* are called *similarity numbers* (or *dimensionless numbers, dimensionless quantities*). The definition of such numbers is based on the theory of invariance and dimensional analysis. The importance of dimensionless numbers comes from the fact that they manifest the effect of the fact that physical laws do not depend on the systems of units that are chosen to represent such laws. E.g., it does not matter whether one uses the SI system of units or CGS, the formulas representing, e.g., Newton's Laws, are the same.

We use the similarity theory methods to the modeling, analysis, and detection of behaviors of multiple UAVs, as a whole treated as a complex dynamical system. The main idea of the similitude theory is that similar behaviors occur when the values of the system variables are in a specific relation. Such relations are captured by dimensionless quantities. Each such relation defines a *hypersurface* in the space spanned over the system variables. Knowing such relationships allows the system to measure the distance to potentially undesirable behaviors, alerting the central controller, which in turn can provide the agents with policies for avoiding such behaviors.

The dimensionless quantities are obtained using the theorem known as Buckingham's *π-theorem* [61]. The *π*-Theorem has two premises and one conclusion. The premises are:

- There exists a function $Y = F(X_1, ..., X_n)$ that relates the variables of interest,
- The function F is invariant with respect to the transformations of systems of units.

The conclusion of the π-Theorem is then that such a function can be represented in dimensionless form $\pi_y = f(\pi_1, ..., \pi_r)$, where the dimensionless variables $\pi_y, \pi_1, ..., \pi_r$ are constructed out of the variables $Y, X_1, ..., X_n$ according to the rules of dimensional analysis. The general form for computing the πs is given by the following formula:

$$\pi_j = \frac{B_j}{A_1^{b_{j1}}, ..., A_m^{b_{jm}}} \tag{14}$$

where A_1, ..., A_m are the dimensional quantities selected from the set X_1, ..., X_n (called dimensional base), while B_1,...,B_r are the remaining quantities from X_1, ..., X_n. Dimensional analysis provides rules and procedures for deriving the πs. Intuitively, the exponents b_{ji} are selected in such a way so that the π_j s are dimensionless.

An even more important implication of the π-Theorem is that it also gives rise to the similitude (or similarity) theory [62]. Similitude theory is used to relate models of

physical reality with respect to the reality, as well as to other models. A number of different requirements need to be satisfied in order for two models to be similar. Most importantly, the values of the dimensionless parameters must be the same for both models in order to achieve similitude between the structures and the behaviors of two modeled systems. These dimensional parameters are called invariants. The reason for this is that whenever these parameters have the same values for two instantiations of the same process captured by the model functions, the behaviors are similar. This provides a quite powerful tool for both prediction and analysis of behaviors. Note that one can modify the specific process variables $X_1, ..., X_n$, yet the values of $\pi_1, ..., \pi_r$ remain constant; the consequence of which is that π_y should also be constant. But since π_y is expressed in terms of some of the $X_1, ..., X_n$ and Y, the value of Y can be predicted. I.e., since we know the value of π_y for the given values of $\pi_1, ..., \pi_r$, we can calculate y as:

$$y = \pi_y \cdot A_1^{b_{j1}}, ..., A_m^{b_{jm}} \tag{15}$$

Additionally, by measuring the values of the variables we can see whether the system stays at the same orbit, where the orbit is defined as the sets of values of the variables X_1, ..., X_n such that the values of all the π_j remain constant. In this way, we can monitor the system for transitions to other orbits, and thus possibly to other types of behaviors. For instance, like in the Reynolds number (*Re*) example, we can see whether the value of $\Re^n$ is moving towards the critical value of 4,000, indicating that the flow may become turbulent, i.e., the system (in this case the process) moves towards a qualitatively different behavior.

The use of dimensionless quantities and similitude theory was studied by Kokar and some results have been published in a number of papers. In [63] the notion of *critical hypersurface* was introduced. Critical hypersurfaces are defined by the dimensionless quantities. The main point of this paper was that hypersurfaces should be used for separating qualitatively different behaviors of systems, rather than hyperplanes. Dimensionless invariants were also used for the purpose of discovery of concepts (relevant variables) that are needed for full characterization of processes.

3.4 Quantitative-Qualitative (Q^2) Approach

While invariants capture the borders between different types of behavior, they do not account for the dynamics of the process behind the transition from one behavior type to another. In a number of papers, Kokar has extended this approach to dynamical systems. In [2], dimensionless invariants were used for monitoring behaviors of time-varying dynamical systems. Then they were also used for learning models of dynamical systems [64, 65, 66].

The notion of consistent quantitative-qualitative dynamic system, or the Q^2 system, was introduced in [67]. The Q^2 approach is used to develop a qualitative dynamic model (QDS) of a quantitative dynamical system (GDS) and then use the qualitative model to analyze the behavior of the dynamical system. While the results of such analysis are qualitative in nature, since intervals and regions are used instead of points, the qualitative conclusion still should be correct. Abstractions that satisfy such requirements are called *consistent*. The Q^2 framework for reasoning with abstractions about a GDS is shown in Figure 12.

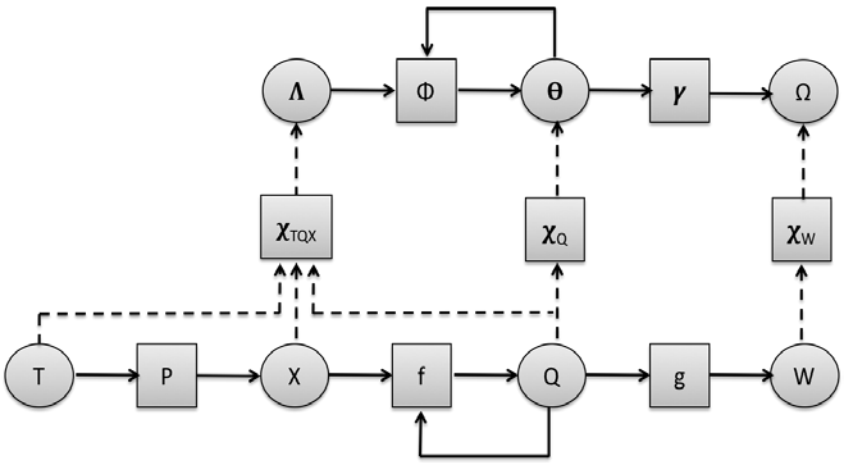


Figure 12. Q^2: Quantitative-Qualitative Representation of a Dynamical System

The consistency constraints on abstractions are defined as [67]:

Let S, Σ and χ represent a GDS, a QDS and an abstraction function. The pair (Σ, χ) is a consistent representational structure of a dynamic system S if the following consistency postulates are fulfilled $\forall q, q_o, x, t$:

$$\gamma(\chi_Q(q)) = \chi_W(g(q))$$
$$\phi(\chi_Q(q_o), \chi_{TQX}(t, q_o, x)) = \chi(f(t, q_o, x)) \quad (16)$$

To summarize, the main ideas of Q^2 are as follows.

1. The $\Re_n$ system space must include not only inputs X, states Q and outputs Y, but also the time between state transitions, T.
2. The partition of $\Re_n$ cannot be done by simply selecting specific values for X, Q and T (this would result in hyperbox partitions), but by hypersufraces in $\Re_n$ defined by appropriate similarity numbers (similar to Re).
3. Qualitative abstraction functions are introduced to partition the output space of a dynamical system into qualitative outputs, state space into qualitative states, and $T \times X \times Q$ into qualitative inputs.
4. Qualitative State Machine (QSM) is defined to abstract the state transitions of the underlying quantitative system by the transitions of the QSM.

This construction was proven to be consistent, i.e., the QSM was proven to be correct with respect to the underlying quantitative system, modulo the abstraction.

4 RESULTS AND DISCUSSION

This section presents the experiments performed to simulate and detect undesirable emergent behaviors. For our simulations, we used NetLogo [60] for agent-based modeling and simulation and MATLAB [68] for analysis of results.

4.1 Boids Flocking

We start the discussion with a very simple and popular example of emergent behavior, i.e., *flocking*. Craig Reynolds in 1986 [3] developed a program named *Boids* to simulate flocking as emergent behavior. The complex behavior of flocking arises from the interaction of individual agents (the boids, in this case) adhering to a set of simple rules. Every agent follows exactly the same set of behavioral rules:

1. **Separation**: At each iteration, each boid makes an adjustment to its velocity if it gets too close to a nearby boid.
2. **Alignment**: For the neighbors inside the vision-range of a boid, it aligns its velocity with the average velocity of its neighboring boids.
3. **Cohesion**: For the neighbors inside the vision-range of a boid, it moves towards the centroid of its neighboring boids.

The boids framework is often used in computer graphics, providing realistic-looking representations of flocks of birds and other creatures, such as schools of fish or herds of animals. Figure 13 shows the snapshot from NetLogo simulation of boids rules. Here, the boids (yellow agents or turtles in NetLogo) are moving in flocks – an emergent behavior. Boids in a flock move with the same velocity (or heading) in space (or environment).

Can we detect such emergence algorithmically? The answer is yes. The detection is pretty direct using similitude theory (see Section 3.3). For instance, consider ten boids moving in space. We create πs (dimensionless quantities) by computing the ratios of boids' velocities, such as, $\pi_k = v_i/v_j, i \neq j$. The plot of all πs versus time (1800 ticks) is shown in Figure 14. Saturation of all πs towards value 1 shows an existence of emergence because at that time the boids in a flock move with the same velocity. The aim here is to show the applicability of dimensional analysis to detect emergence in such a simple scenario. We use the same kind of approach to a more complicated scenario of UAVs (explained in the following sections).

4.2 Agent Based Modeling and Simulation of UAVs

In this subsection, we describe the design of control policies for single and multiple UAVs per the persistent surveillance scenario presented in Section 3.2.

4.2.1 Single UAV representation. Figure 15 shows one UAV agent (red plane) in Net- Logo. The environment (2D space) is divided into cells such that each cell has its own coordinate location. We assume that one UAV occupies one cell. Each cell has an associated age value that represents the time elapsed since the last observation (Section 3.2.1). The blue region (Fig. 15)

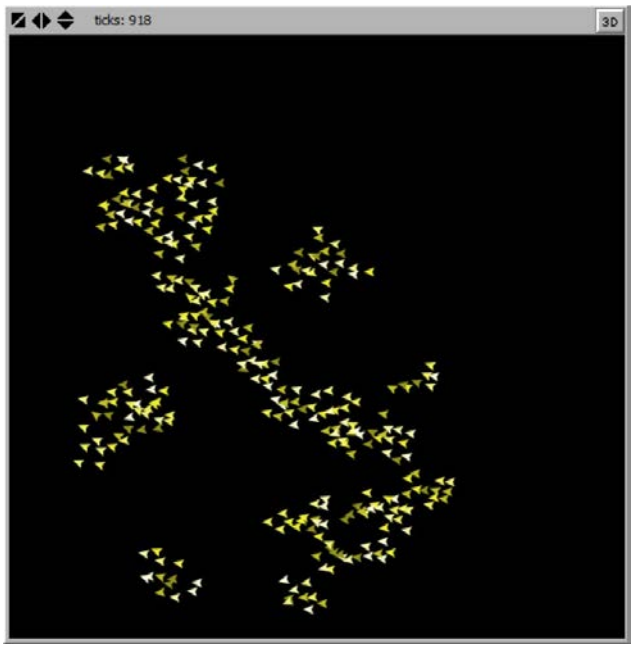


Figure 13. NetLogo Simulation - Boids Flocking

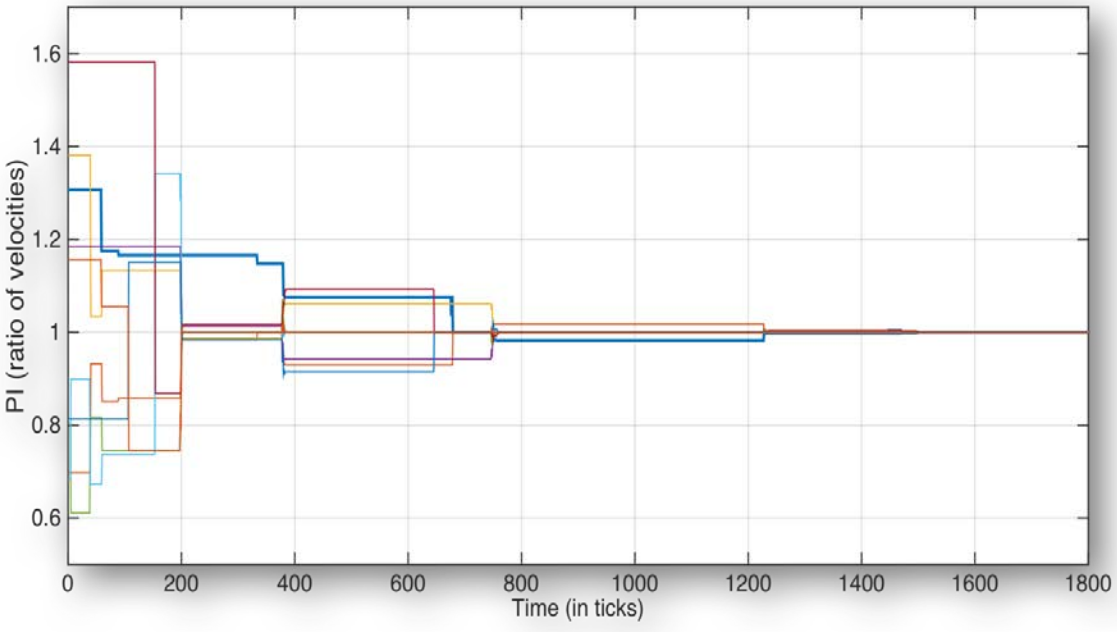


Figure 14. Detection of Flocking Behavior using Similitude Theory

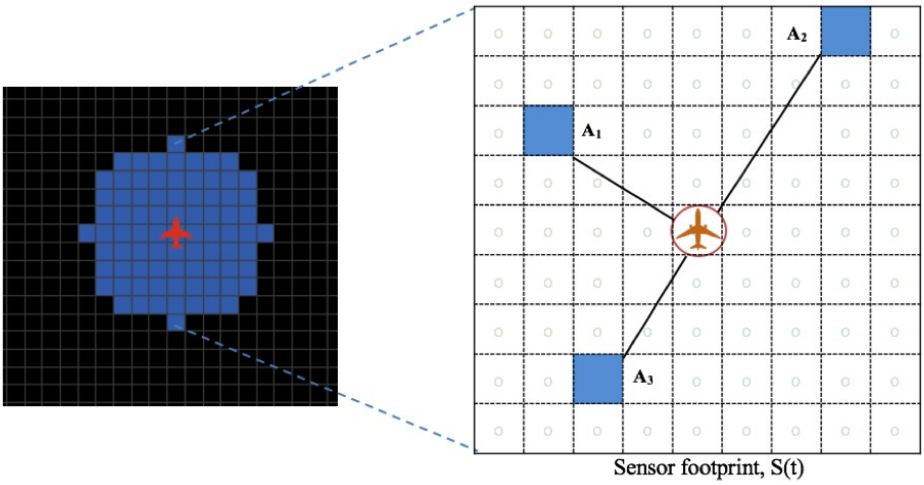


Figure 15. UAV and Information Age

shows the sensor footprint of the UAV which comprises the cells in its sensing range (a constant value). Each UAV is modeled as a point in space with location, velocity and acceleration (force). The acceleration is the input applied in any direction in an effort to steer the UAV. The basic rule for UAV movement is *"seek"* which causes a UAV to move towards some specified (desired) position in the environment.

A UAV agent in our simulations is designed to be a self-controlling autonomous agent. The search for a solution to the optimization problem (Eq. 7) is implemented in the controller of a UAV. At every iteration (or tick NetLogo), each UAV selects a value of the acceleration based on its current state (position, direction and velocity) and the age values of cells in its sensing range (neighboring cells). Thus, the control policy for a single UAV is:

> *Measure the surroundings (sensing range), create feedback and calculate next location to move to, based on nearest maximum age cell.*

Figure 15 gives the pictorial representation in one-dimensional space. The next maximum age cell (i) is selected based on the following equation:

$$i = \arg\max_{j \in N} A_j, \quad A_j - age\ of\ cell\ j, N - indexes\ of\ the\ cells \tag{17}$$

In our simulations, we observed different behaviors of the UAVs moving in straight and curved lines. One of the interesting emergent behaviors observed was the "lawn-mowing" pattern of the movements of the UAVs. Figure 16 shows a MATLAB simulation output for a single UAV implementation of the square plume monitoring. With no dynamic constraints on the rate of turn, the typical lawnmower behavior emerges, while a random turn radius is allowed, the UAV's behavior does not exhibit such a pattern.

4.2.2 Extension to multiple UAVs. In the previous subsection, we described a control policy for one UAV performing a specific mission (plume monitoring). In this subsection, we

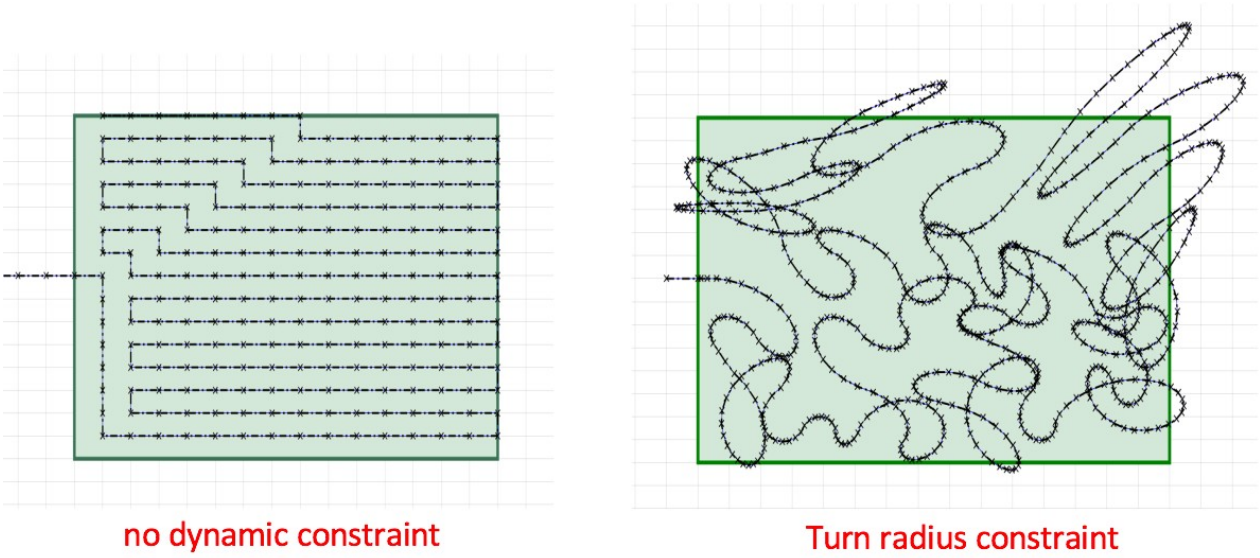


Figure 16. Pattern Emergence from Single UAV (Plume - Green Area)

extend the control policies to multiple UAVs. As explained in Section 3.1.1, for a streamlined movement and interactions among UAVs, we made each UAV to follow four steps of the OODA loop. A UAV agent takes appropriate action based on its observation and orientation. At any time, the action of a UAV agent is based on the following information:

- *Goal Space*: The goal in this scenario is to minimize the information age metric i.e. to solve the optimization problem (Eq. 7).
- *Action Space*: Each UAV calculates its next location by measuring its sensor footprint
- *Constraints Space*: For UAV-UAV interaction, we follow three constraints/ behavioral rules from [3] (pictorial representation shown in Fig. 17):
  - Separation Constraint: Maintain minimum distance with neighboring UAV (neighborhood is defined by sensor footprint),
  - Alignment Constraint: Heading of each UAV is equal to the average heading of the neighboring UAVs,
  - Cohesion Constraint: Each UAV moves in the direction of the centroid of the neighboring UAVs.

The aim here is to simulate this multi-UAV system and study different behaviors emerged at the system level. Section 4.1 shows that three simple rules cause the flocking to appear. The question here is, are we going to get the same emergent behavior with UAVs? The easiest and fastest approach is by simulation. We observe what happens at the system (or macro) level when UAVs interact at the micro level (Section 3.1.1). The algorithms for agent and system logic are given in Algorithm 1 and 2.

Figures 18 and 19 show the NetLogo snapshots for a 4-UAV system implementation. The sliding bars on the left are for the external parameters that can be changed during the

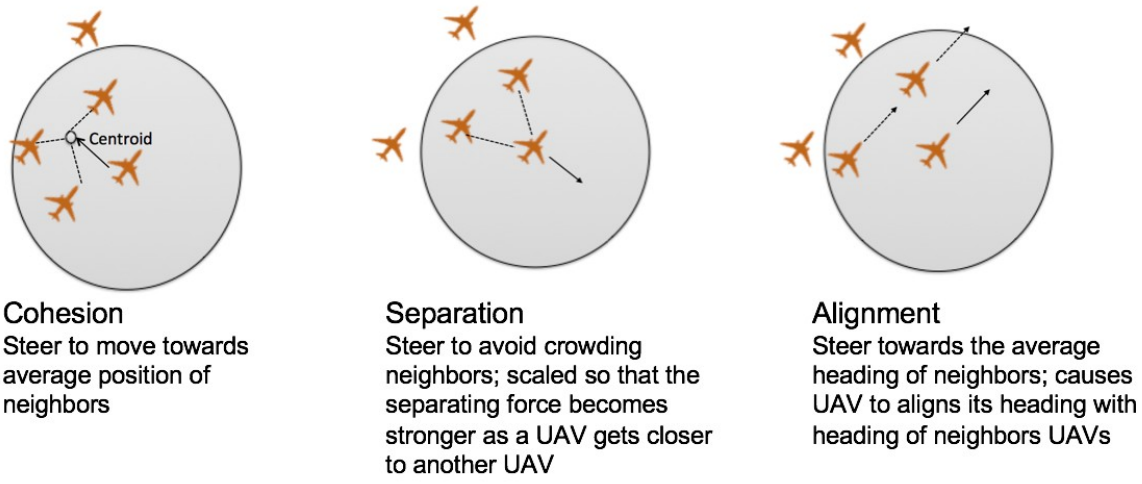


Figure 17. Control Policies for Multiple UAVs (grey circle represent the $S(t)$ of center UAV

Algorithm 1 Agent Logic

```
procedure AGENTSTEP
    nearest_neighbor ← FindNearestNeighbor(Agent_Position)
    max_age_cell ← FindMaxAgeCell(Agent_Position, Agent_VisionRange)
    distance ← CalculateDistance(Agent_Position, nearest_neighbor_Position)
    if distance <= separation_threshold then
        new_heading ← MoveAway(Agent_Heading, nearest_neighbor_Heading)
    else
        new_heading ← MoveTowards(Agent_Position, max_age_cell)
    end if
     Move(new_heading)
end procedure
```

Algorithm 2 System Logic

```
procedure SYSTEMSTEP
    for all agents do
        S ← SensorFootprint(Agent_Position)        ▷ Locations of cells
        UpdateCellAge(S, 0)                        ▷ Set the age of S to zero
    end for
    I_age ← Sum(CellAge)                           ▷ sum of all cells
end procedure
```

simulation. The plot on the right represents information age over time. We simulated the

scenario with two kinds of plumes (the green region in the figures) - Square and Circular. The simulations are not affected by the shape of the plume, thus, for further results, we just show the runs with the circular plume. Also, to note here, in all the simulations, we used plumes with the same concentration in the plume, i.e., all cells are the same.

In this work, our aim is to simulate and study undesirable emergent behaviors in swarms of UAVs. Figures 20 and 21 demonstrate two cases of undesirable emergent behaviors in a persistent surveillance scenario. Figure 20 shows the undesirable flocking of UAVs due to the alignment and cohesion constraints. This kind of emergent behavior is undesirable in this scenario as it causes the quality metric, i.e., information age, to increase. Figure 21 shows the monitoring of two plumes by 4 UAVs. The simulation clearly shows the *poorly covered or non-covered facility* emergent behavior (Section 2.5). Again, this behavior is undesirable as none of the UAVs are monitoring the first plume, causing an increase in the overall value of the information age.

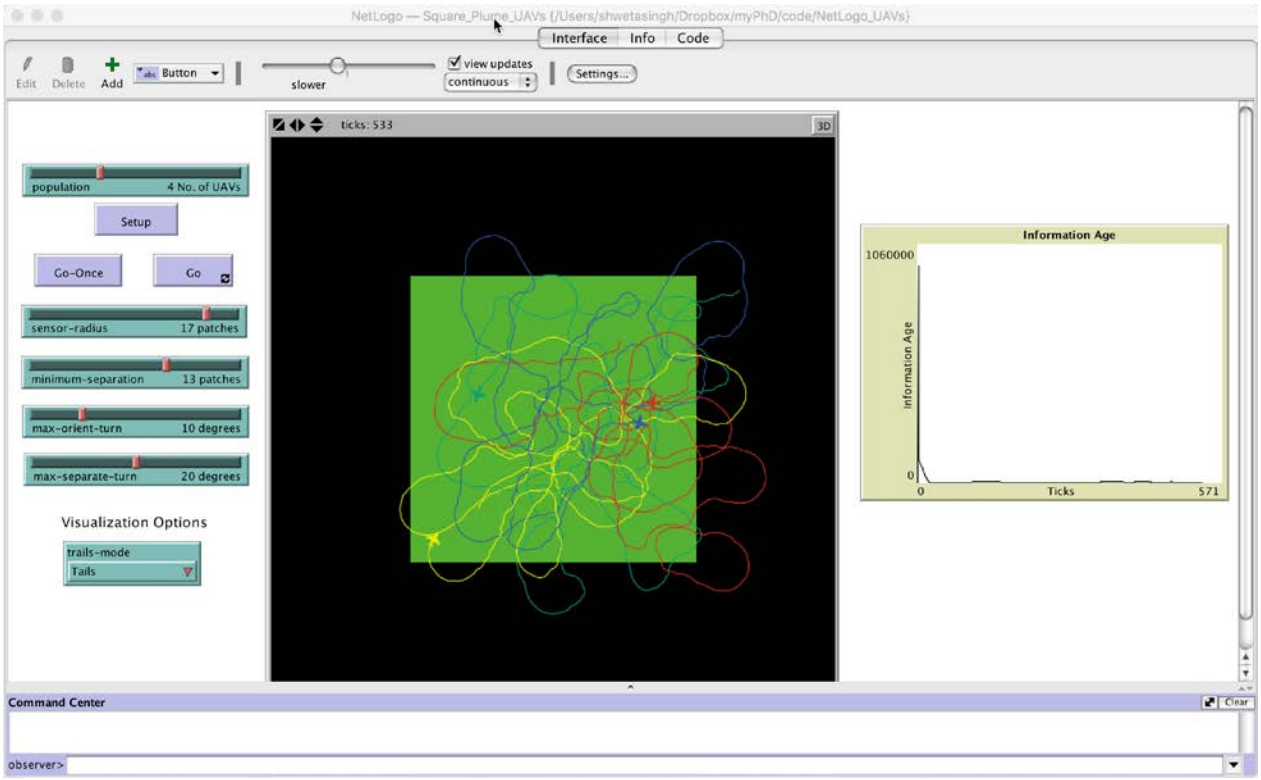


Figure 18. NetLogo Simulation of persistent surveillance of square plume by 4 UAVs

4.2.3 Simulating Emergent Behavior Types. Previous sections presented some of the examples of desirable and undesirable emergent behaviors in swarms of UAVs. But so far, we have not distinguished these behaviors into different types. This subsection deals with simulating different types of emergent behaviors in the multi-UAV system. The main idea here is to simulate the behaviors such that they fall into the specified types of emergence.

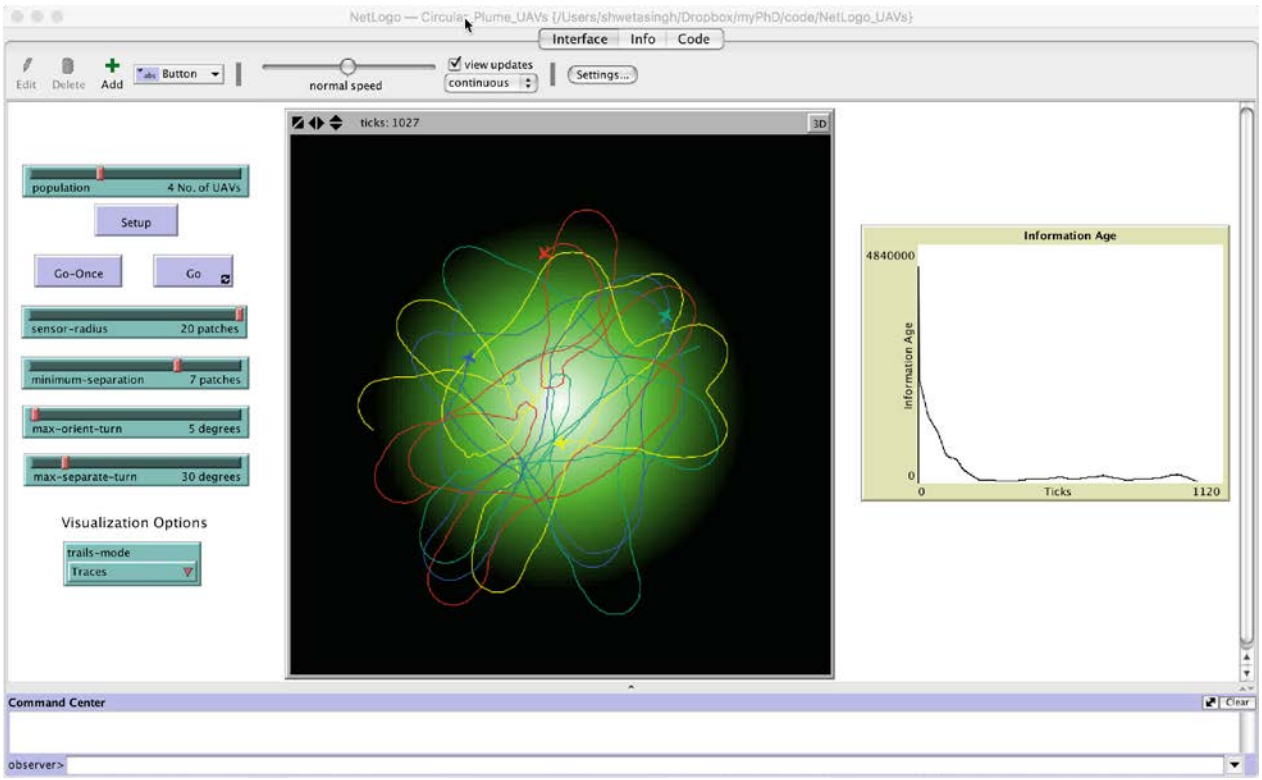


Figure 19. NetLogo Simulation of persistent surveillance of circular plume by 4 UAVs

As mentioned above, we follow the classification provided by Fromm (Section 2.4.1). Here, we discuss two types i.e., type IIa and type IIb as these are the most interesting with respect to engineering applications.

1. **Type IIa:** This refers to the emergent behavior type in which the net feedback from the macro level is negative. For the discussed UAV scenario, this is implemented using the information age metric, i.e., each UAV's motion is constrained by the metric of information age in such a way that it has to move towards the cell with highest age value at each time instant. Figure 22 shows the simulation where UAVs interact by only following the separation constraint. This results into a desired behavior of the UAVs as the information age decreases with time and stays minimum over time.

2. **Type IIb:** This refers to the type where the net feedback from the macro level is positive. For the discussed UAV scenario, we implemented this by adding alignment and cohesion constraints. This results into group formation (Fig. 23), as in boids, which turns out to be an undesirable behavior in UAVs as it causes information age to increase with time.

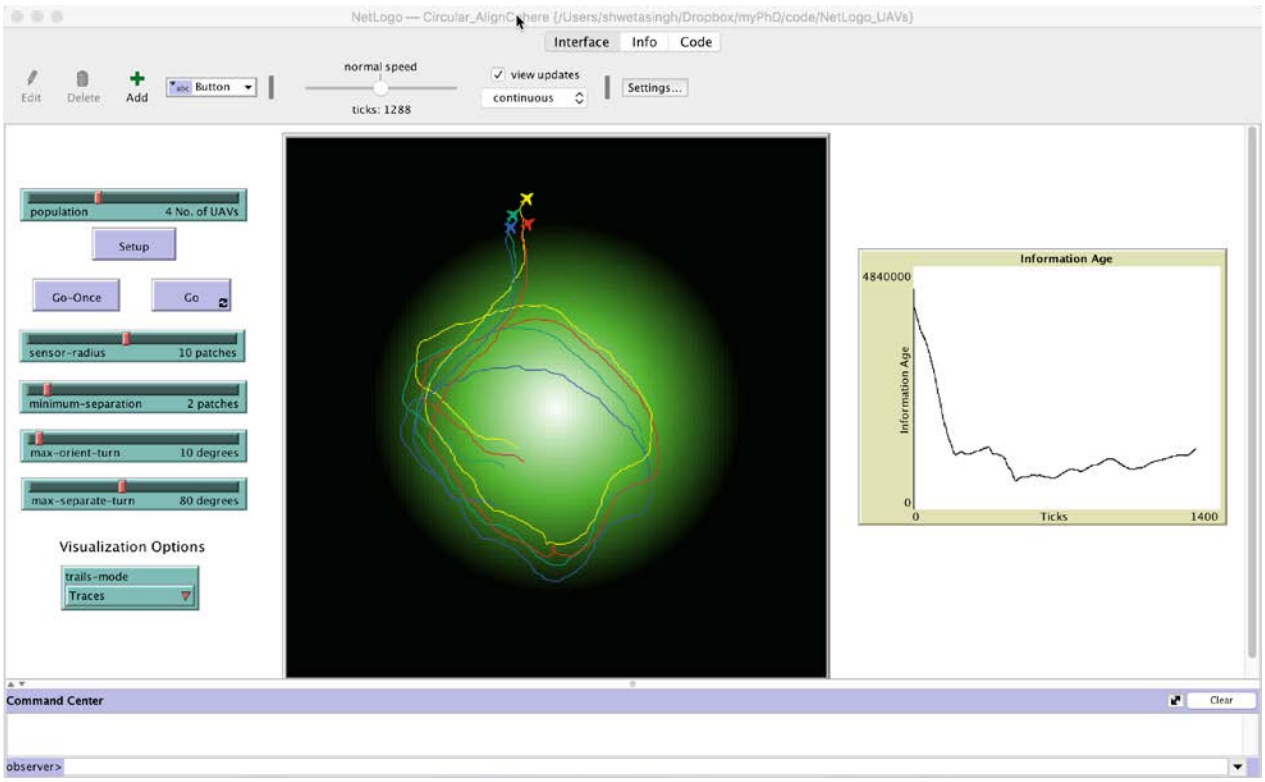


Figure 20. Flocking Emergent Behaviors in Multi-UAV Plume Monitoring System.

4.3 Application of Q^2 Approach

This section presents the step by step application of the Quantitative-Qualitative (Q^2) approach. We start the discussion with a simple test case scenario with two UAVs and then formalize a generalized Q^2 conceptualization for a dynamical system.

4.3.1 A Test Case Scenario. The Q^2 approach translates observations of the dynamical system to qualitative inputs, outputs, and states. As a first step in this direction we implement a test case scenario with two UAVs moving in circular motion, one clockwise and another counterclockwise (shown in Fig. 24). The system of these two UAVs is treated as one dynamical system. We observed their synchronized motion using qualitative outputs and tried to learn the hypersurfaces they are moving through.

Considering heading (α) as the output of each UAV, the ranges for α_1 and α_2 are $[0^o, 360^o)$ (counterclockwise direction) and $[180^o, -180^o)$ (clockwise direction), respectively. Figure 25 shows the observation of outputs of these two UAVs. The quantitative model, i.e. states (Q), inputs (X), state transition function (f) and output (W) sets are given below:

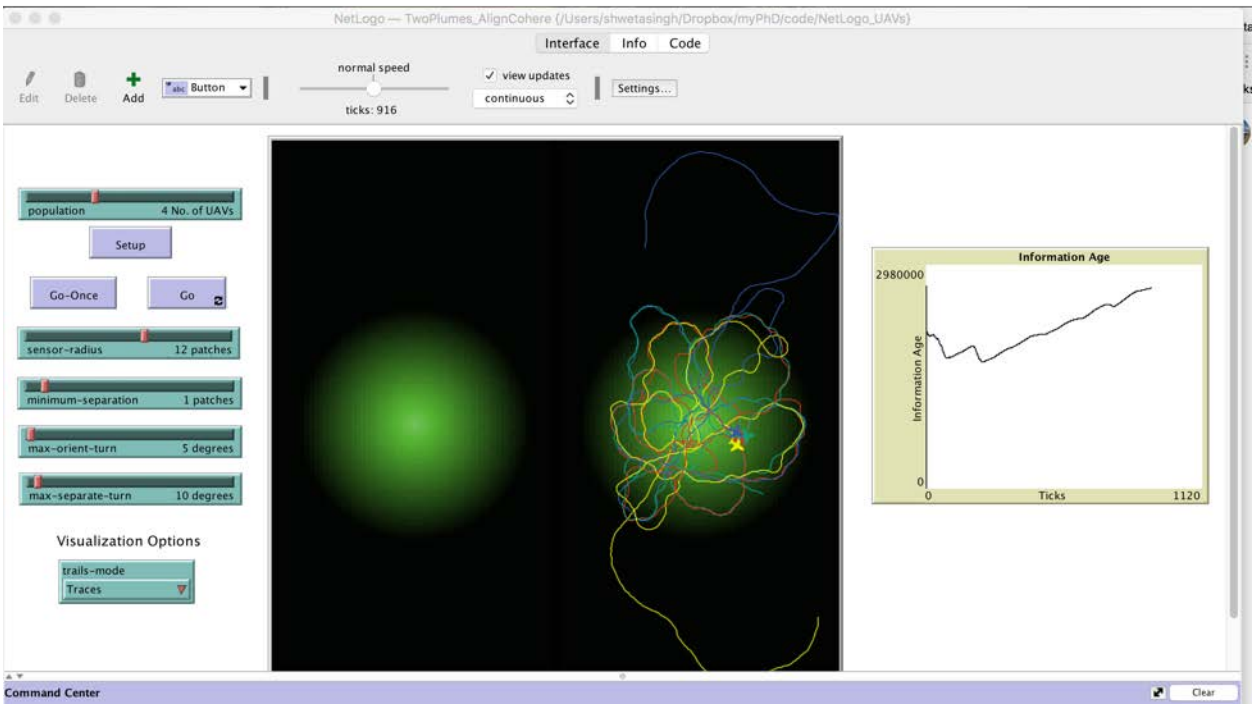


Figure 21. Undesirable Emergent Behavior - Non-Covered Facility

$$\begin{aligned}
\mathbf{q}(k) &= [x_1(k), y_1(k), v_{x1}(k), v_{y1}(k), x_2(k), y_2(k), v_{x2}(k), v_{y2}(k)]^T \\
\mathbf{x}(k) &= [a_{x1}(k), a_{y1}(k), a_{x2}(k), a_{y2}(k)]^T \\
\mathbf{q}(k) &= \boldsymbol{f}(\mathbf{q}(k-1), \mathbf{x}(k-1), k) \\
\mathbf{w}(k) &= [\alpha_1(k), \alpha_2(k)]^T
\end{aligned} \tag{18}$$

To create the qualitative model for this system, we start with qualitative partitioning of the output space. Let us assume eight (8) partitions of the output, i.e., output abstraction function divides the output space into eight qualitative outputs (more details in the next section):

$$\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8\} \tag{19}$$

Figure 26 shows a trajectory through such qualitative output space. The lines show the movements of the UAVs in time (quantitative). The blue and red boxes in the plot represent the points that fall inside the qualitative partitions of the Ω space.

This kind of behavior generates a hypersurface in space. The line going through the coordinate plot in Figure 27 represents the hypersurface for this dynamical system. This is obtained by determining the invariance in outputs of these two agents. α_1 and α_2 are related by the equation, $\alpha_2 = \alpha_{20} - \alpha_1$, where α_{20} is initial value for α_2. It means that as long as both the UAVs are moving along this hypersurface, there will be a synchronized motion (circular motion in different directions in this case), making it easier to detect how agents are moving in space. Thus, our ultimate goal is to generate or learn such hypersurfaces for our specified multi-UAV system. Such hypersurfaces divide the space into distinct behaviors which can be used to separate desirable behavior from undesirable ones.

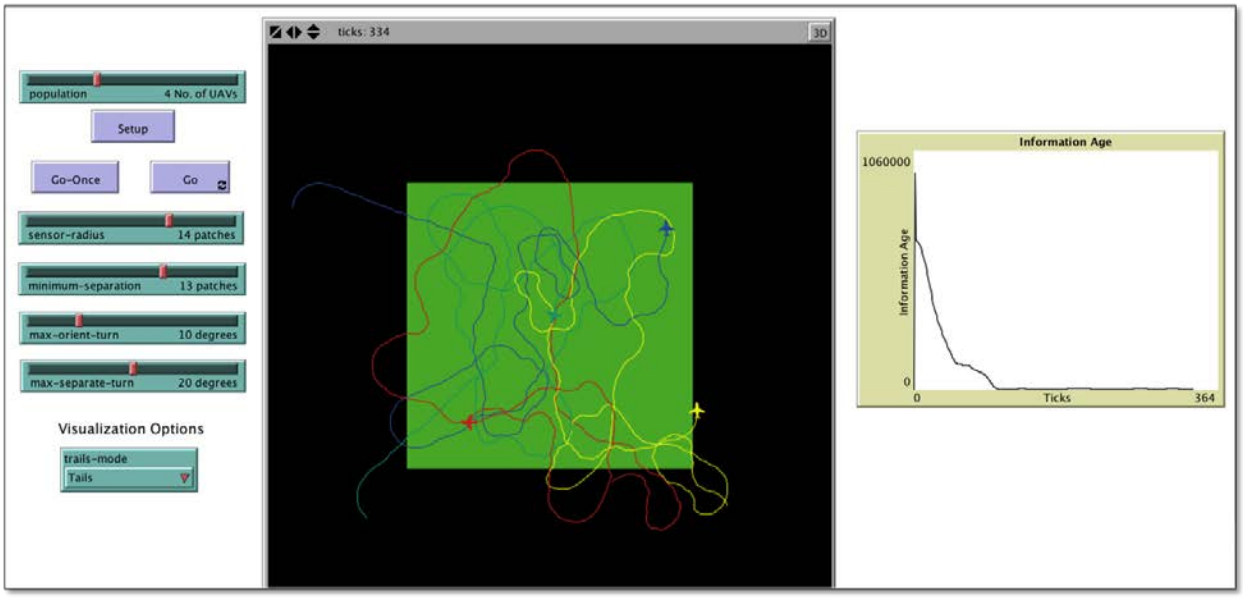


Figure 22. NetLogo Simulation of Persistent Surveillance by 4 UAVs of Square Plume (Green Region) - Minimization of Information Age (Right Plot)

4.3.2 Generalized Q^2 Conceptualization. In this section, we develop the Q^2 conceptualization for a UAV agent. As mentioned before, we consider UAV as a dynamical system with system dynamics given in Section 3.2.2. We use the same state transition function f as in Equation 9. Let's take the heading direction, $\alpha(k)$, as the output variable for the system, which is defined as:

$$\alpha(k) = \begin{cases} \arctan(v_y(k)/v_x(k)) & v_x(k) \geq 0, v_y(k) > 0 \\ \arctan(v_y(k)/v_x(k)) + \pi & v_x(k) < 0, v_y(k) \geq 0 \\ \arctan(v_y(k)/v_x(k)) + \pi & v_x(k) < 0, v_y(k) < 0 \\ \arctan(v_y(k)/v_x(k)) + 2\pi & v_x(k) \geq 0, v_y(k) < 0 \end{cases} \tag{20}$$

We intentionally ignore the case where $v_x(k) = 0$ and $v_y(k) = 0$ as it is assumed that agent won't stop moving in the space.

4.3.2.1 Partitioning of the Output Space and Qualitative Outputs. We start developing qualitative abstractions with the output space. In this case, the output space is $W = [0, 2\pi)$ in radians, or $[0^o$ to $360^o)$ in degrees. The qualitative outputs (sets in the original space) are separated by output landmarks:

$$\{\alpha_1^* = 22^o, \alpha_2^* = 67^o, \alpha_3^* = 112^o, \alpha_4^* = 157^o, \alpha_5^* = 202^o, \alpha_6^* = 247^o, \alpha_7^* = 292^o, \alpha_8^* = 337^o\} \tag{21}$$

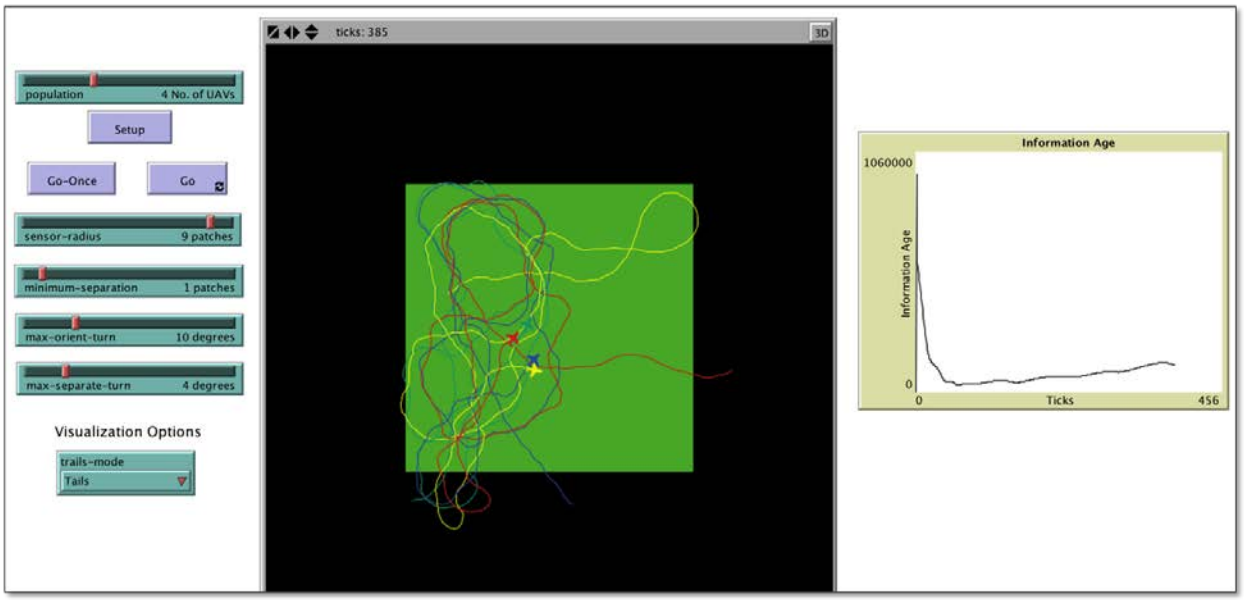


Figure 23. NetLogo Simulation of Persistent Surveillance by 4 UAVs of Square Plume (Green Region) - Undesirable Group Formation

The qualitative outputs include eight elements:

$$\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8\} \tag{22}$$

Thus, the output abstraction function is given by the following equation:

$$\chi_W(\alpha) = \begin{cases} \omega_1 & \alpha \in 0^o \pm 22^o \\ \omega_2 & \alpha \in 45^o \pm 22^o \\ \omega_3 & \alpha \in 90^o \pm 22^o \\ \omega_4 & \alpha \in 135^o \pm 22^o \\ \omega_5 & \alpha \in 180^o \pm 22^o \\ \omega_6 & \alpha \in 225^o \pm 22^o \\ \omega_7 & \alpha \in 270^o \pm 22^o \\ \omega_8 & \alpha \in 315^o \pm 22^o \end{cases} \tag{23}$$

As can be seen from Equation 23, the output abstraction function χ_W defines eight (disjoint) partitions or subsets in the quantitative output space.

4.3.2.2 Partitioning of State Space and Qualitative States.

According to the Q^2 approach, the partitioning of the output space determines qualitative partitions in the state space $X \times Y \times V_X \times V_Y$. The goal is to obtain the qualitative state abstraction function, χ_Q, of the state space **Q** such that

$$\chi_Q : \mathbf{Q} \rightarrow \Theta \tag{24}$$

where each qualitative state θ_i corresponds to one qualitative output ω_j.

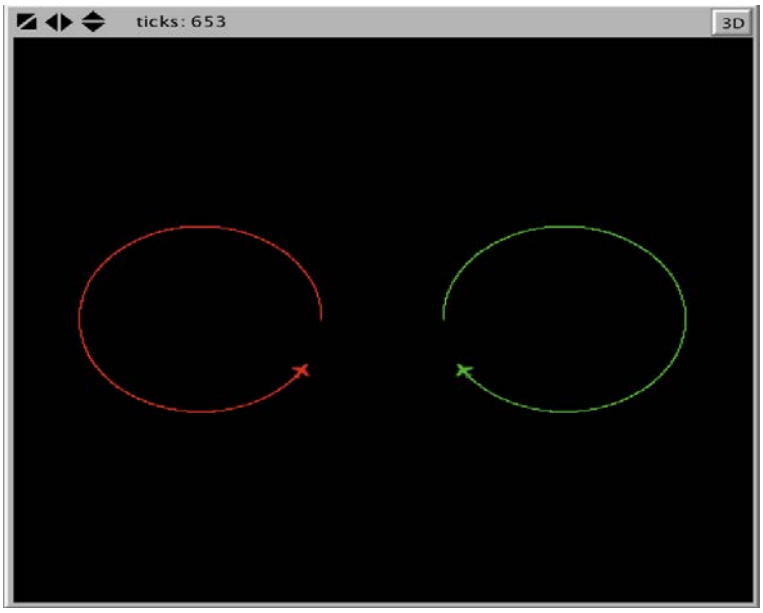


Figure 24. Two UAVs Moving in Circular Motion (Opposite Directions)

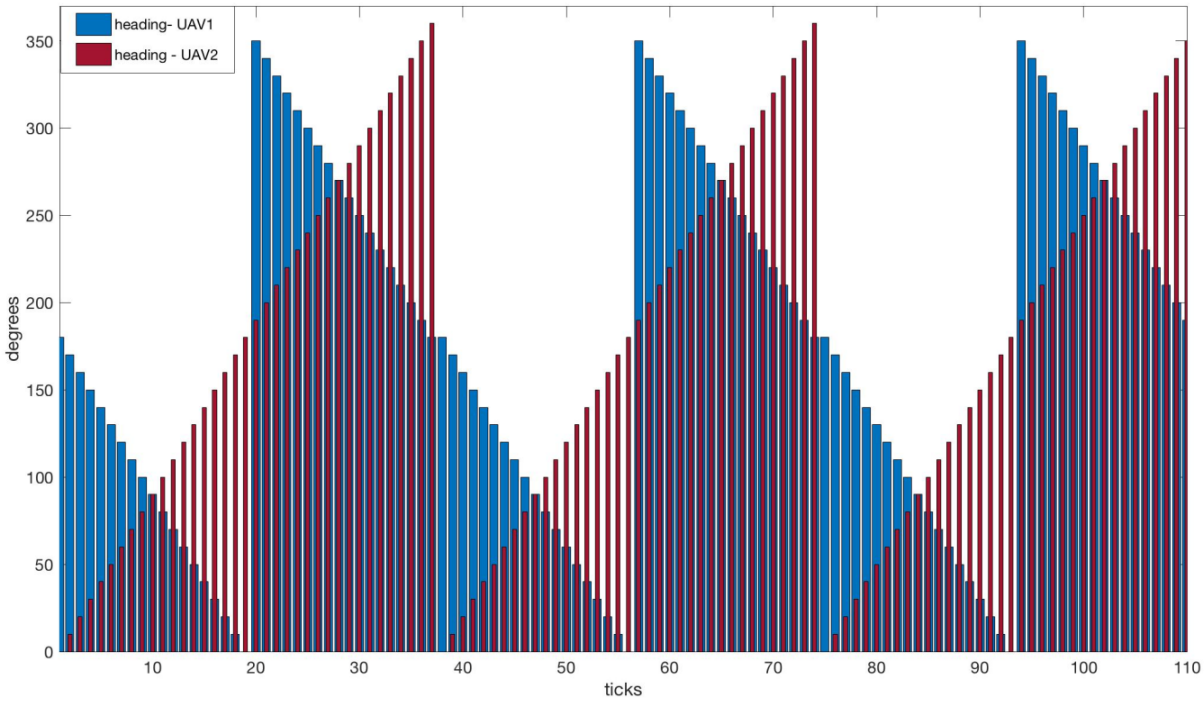


Figure 25. Observation of Outputs

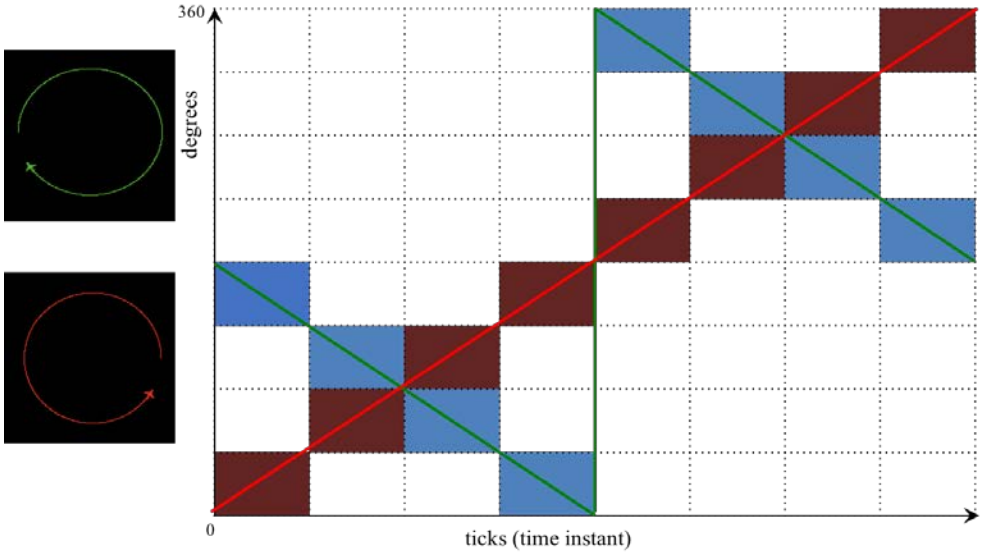


Figure 26. Transition through Qualitative Space

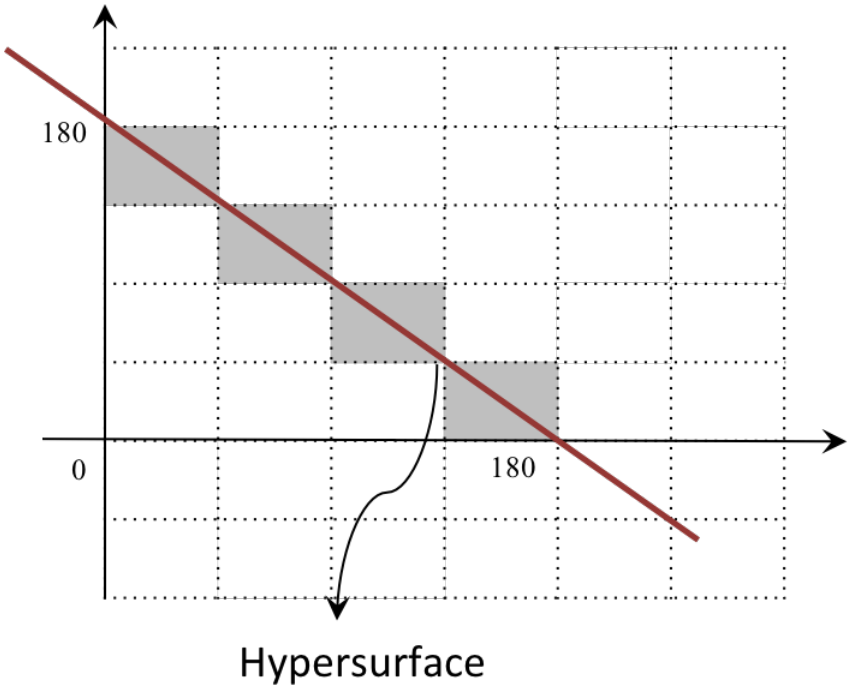


Figure 27. Invariance in Outputs

The eight qualitative outputs induce the partition of the state space into eight regions (qualitative states) $\{\theta_1, ..., \theta_8\}$, each corresponding to a qualitative output.

$$\Theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8\} \tag{25}$$

The qualitative output function γ is defined in such a way that θ_1 corresponds to ω_1, θ_2 corresponds to ω_2, ..., θ_8 corresponds to ω_8. The explicit assignment is represented as:

$$\begin{aligned}\gamma = \{&< \theta_1, \omega_1 >, < \theta_2, \omega_2 >, < \theta_3, \omega_3 >, < \theta_4, \omega_4 >, \\ &< \theta_5, \omega_5 >, < \theta_6, \omega_6 >, < \theta_7, \omega_7 >, < \theta_8, \omega_8 >\}\end{aligned} \tag{26}$$

The computation of χ_Q is achieved by the following steps:

1. For each state $q \in Q$, compute the output associated with q by the output function α (it is denoted as g in Fig. 12).

2. Find the qualitative output corresponding to this output by the qualitative output function χ_W.

3. Find the qualitative state $\theta \in \Theta$ that is mapped to this qualitative output by the qualitative output function γ. Assign this qualitative state to q.

This computation defines χ_Q, which can be expressed mathematically as:

$$\chi_Q(q) = \gamma^{-1}(\chi_W(\alpha(q))) \tag{27}$$

An example of projection of state space partition onto 2D is shown in Figure 28.

To test whether the two abstraction functions (χ_W and χ_Q) define a consistent abstraction, we can test the satisfaction of the consistency constraints between general dynamic system and qualitative dynamic system with respect to state abstraction, given by Equation 28, by performing computation in the forward manner, i.e., testing whether for each state q, the mapping from q to Θ and then applying the qualitative output function γ gives the same result as first applying the output function α to the state q and then mapping the quantitative output to the qualitative output using the qualitative output abstraction χ_W, as shown in Equation 28.

$$\gamma(\chi_Q(q)) = \chi_W(\alpha(q)) \tag{28}$$

4.3.2.3 Partitioning of Input Space and Qualitative Inputs. The qualitative input abstraction function χ_{TQX}, of **TQX** space, is obtained in a similar way as the abstraction function for the states. We denote the qualitative input set as Λ, such that:

$$\chi_{TQX} : \mathbf{TQX} \rightarrow \Lambda \tag{29}$$

We postulated that the qualitative input set consists of eight qualitative inputs as shown in Equation 30:

$$\Lambda = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7, \lambda_8\} \tag{30}$$

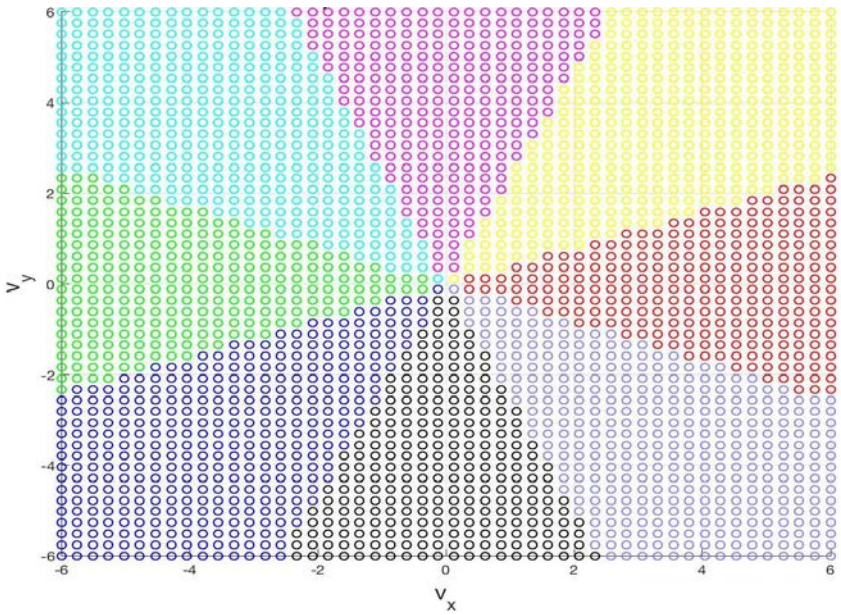


Figure 28. Partitioning of State Space [Red: θ_1, Yellow: θ_2, Magenta: θ_3, Cyan: θ_4, Green: θ_5, Blue: θ_6, Black: θ_7, Light Blue: θ_8]

We assume that λ_1 causes transition to θ_1, λ_2 causes transition to θ_2 and so on. This defines the qualitative state transition function, ϕ, represented as an automaton shown in Figure 29. The automaton is a two-way loop in which the heading direction of the particle in the space is limited in the range $[0, 360)$. For instance, if particle is moving in a clockwise direction along a circular path and is in state θ_2 it can change its path to counterclockwise direction by applying the input λ_1 resulting into transition to state θ_1.

Similarly, as with the computation of the qualitative state abstraction, we compute the qualitative input abstraction function χ_{TQX} in the following steps:

1. For each quantitative (initial) state $q_0 = (x, y, v_x, v_y)$, quantitative input $u = (a_x, a_y)$ and time increment T: compute the next state $q' = (x', y', v_x', v_y')$.
2. Compute the qualitative abstractions of the initial state and the next state.
3. Find the qualitative next state, θ_i and assign its label to the qualitative input, λ_i.

The computation of the qualitative input abstraction function, χ_{TQX}, can be described mathematically as:

$$\chi_{TQX}(q_o, u, T) = \phi^{-1}(\chi_Q(q_0), \chi_Q(f(q_0, u, T))) \tag{31}$$

As an example of input partitions, a projection onto 2D is shown in Figure 30. The horizontal axis is time, T, while the vertical axis represents the y-component of the acceleration input, a_y.

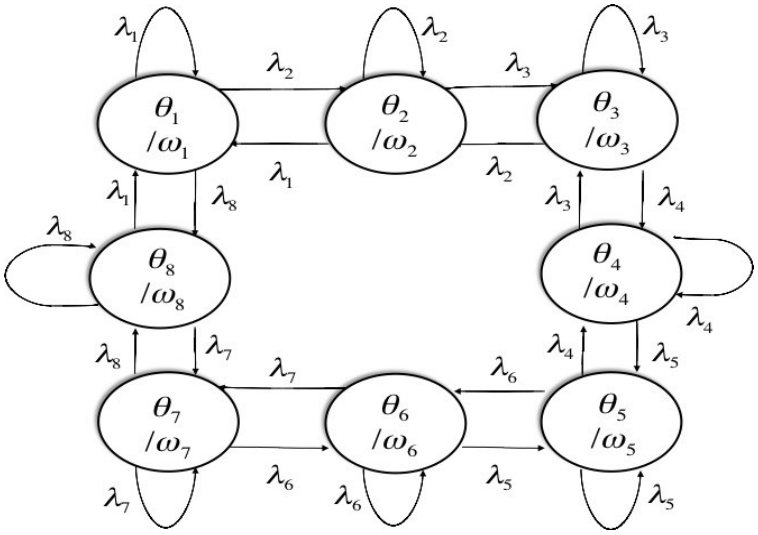


Figure 29. Automaton: Qualitative State Transition Function

In order to be a consistent abstraction function, the qualitative input abstraction function χ_{TQX} should satisfy the consistency constraint specified in Equation 16. To ensure that our implementation of the qualitative input abstraction of Equation 31 is correct, we test the satisfaction of this condition by performing the forward computation defined by Equation 32.

$$\phi(\chi_Q(q_0), \chi_{TQX}(q_0, u, T)) = \chi_Q(f(q_0, u, T)) \tag{32}$$

4.3.3 Proof of consistency constraints.

1. *Qualitative State Abstraction*

   Given a state, $q \in Q$, we need to prove that the following two steps (LHS and RHS of Eq. 28) result in the same qualitative output:

   1.1. Map q to Θ using $\chi_Q \Rightarrow \theta_i$, then apply qualitative output function γ on $\theta_i \Rightarrow \omega_i$

   1.2. Map q to W using α (quantitative output function) and then apply χ_W to get qualitative output $\Rightarrow \omega_j$

   Thus, we need to prove that $\omega_i \equiv \omega_j$. We prove this graphically, Figure 31 shows the plots for above steps. It is clearly visible that both result into same partition of the state space. Each point is assigned a color representing the qualitative output it belongs to (based on the computations described in above steps).

2. *Qualitative Input Abstraction*

   As discussed in above sub-section, we prove the consistency constraint on input abstraction function by proving that following two steps are equivalent (LHS and RHS of Eq. 32). Given initial state, q_0, quantitative input, u, and time, T:

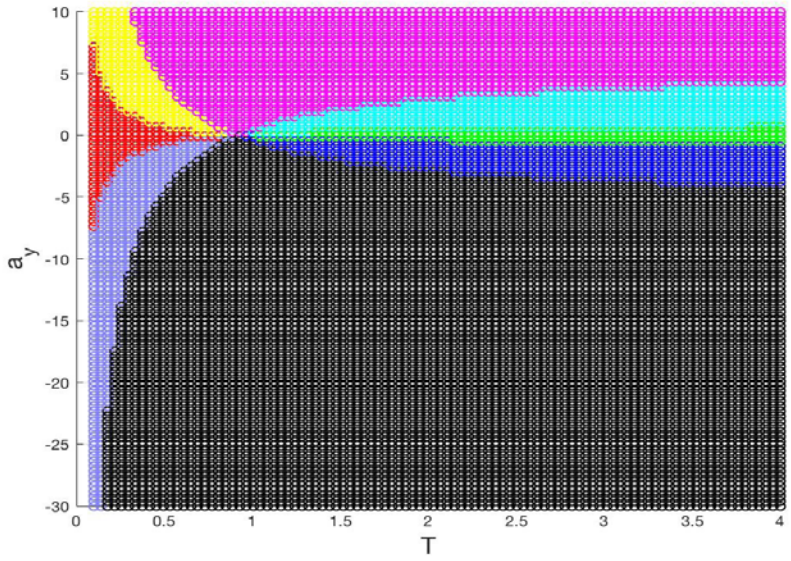
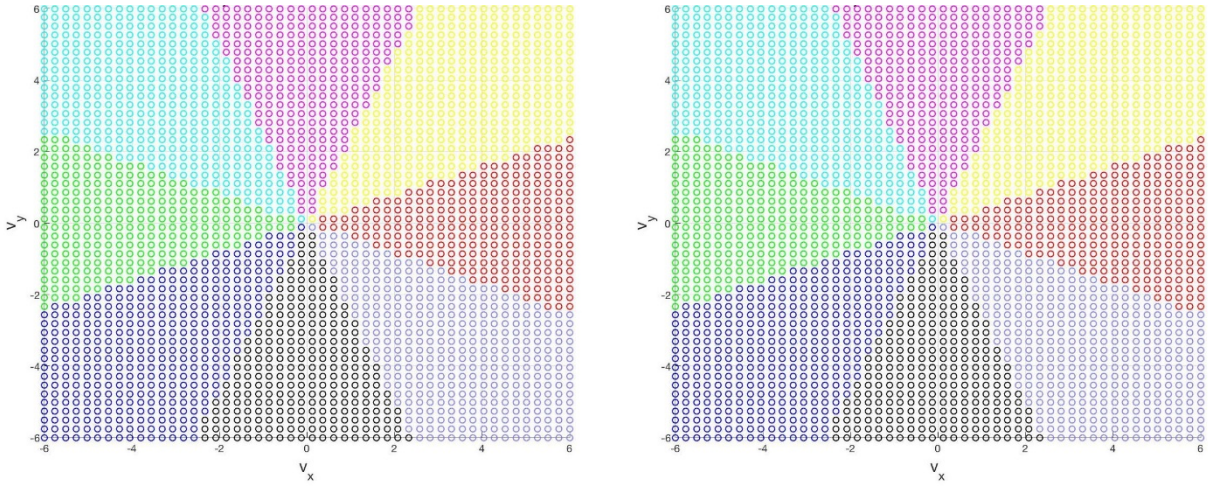


Figure 30. Partitioning of Input Space [Red: λ_1, Yellow: λ_2, Magenta: λ_3, Cyan: λ_4, Green: λ_5, Blue: λ_6, Black: λ_7, Light Blue: λ_8]



(a) Using Step (1) **(b) Using Step (2)**

Figure 31. Consistency Constraint [Red: ω_1, Yellow: ω_2, Magenta: ω_3, Cyan: ω_4, Green: ω_5, Blue: ω_6, Black: ω_7, Light Blue: ω_8]

2.1. Apply χ_Q on q_0 to get initial qualitative state and χ_{TQX} on $\{q_0, u, T\}$ to get qualitative input and then apply φ to get the next qualitative state $\Rightarrow \theta_i$

2.2. Apply state transition function, $\boldsymbol{f}$, on $\{q_0, u, T\}$ to get the next quantitative state and then apply χ_Q to get its qualitative equivalent $\Rightarrow \theta_j$

Thus, we need to prove that $\theta_i \equiv \theta_j$. This is done in similar way as in previous subsection.

4.3.4 Simulation in Qualitative Domain. This section presents the simulation results based on the qualitative conceptualization developed in previous subsection. The steps for performing reasoning using Q^2 are given below:

1. Determine current qualitative state using χ_Q
2. Select the desired next qualitative state
3. Find the qualitative input (λ_i) according to automaton that causes the transition to the desired state
4. Select a (quantitative) value for the input vector from the region associated with λ_i

Figure 32 demonstrates two types of emergent behaviors - Type IIa and Type IIb. The left part shows the simulation of two UAVs for the Type IIa emergent behavior with top-down negative feedback (constrained behavior due to information age). This is a desirable behavior as the information age decreases over time. On the other hand, the right part shows the Type IIb emergent behavior where two UAVs try to align with each other resulting into undesirable behavior with increased information age.

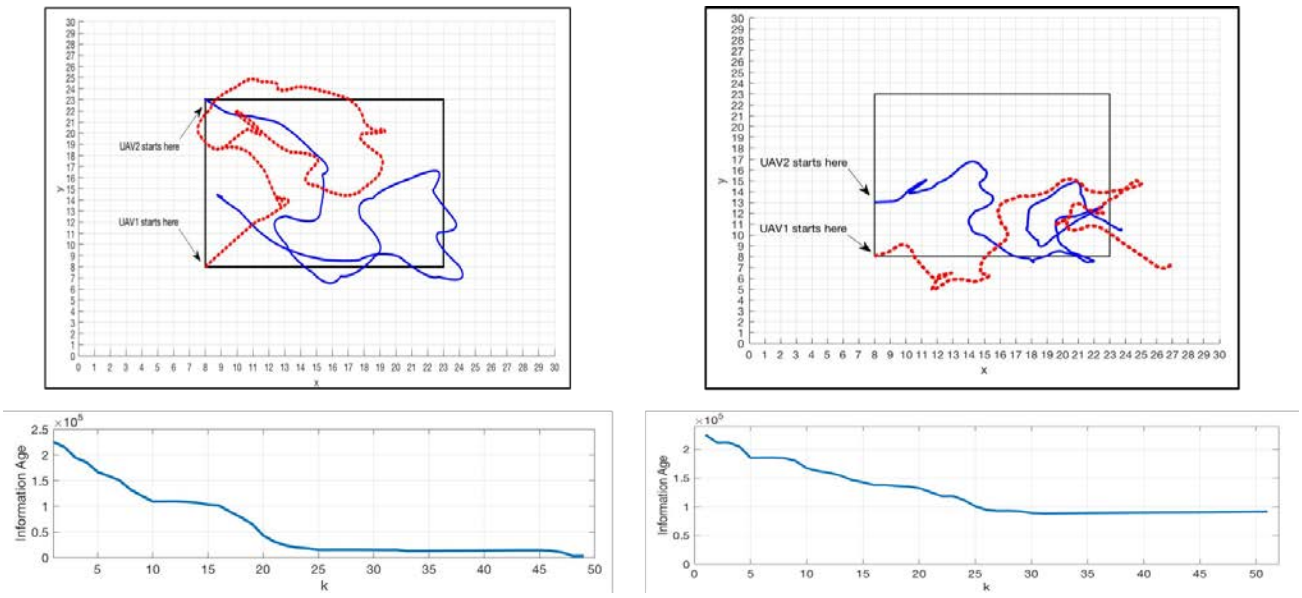


Figure 32. MATLAB Simulation of Type IIa (left) and Type IIb (right) Emergent Behavior using Q^2

4.4 Detection and Classification of Emergent Behaviors

4.4.1 Detection using Variety. So far, in our experiments, the existence of undesirable emergent behavior was confirmed by studying the changes in the quality metric (information age) of the scenario. Also, the simulation platforms, like NetLogo, provide just visual inspection of emergence using its built-in animator window. We need a mechanism for automatic detection of emergence. Literature provides many alternatives for detection of emergent behaviors mathematically. These approaches are categorized as variable-based, grammar based and event-based techniques. In our research, we use the variable-based approach, i.e. using a variable or a metric to prove the existence of emergence. In particular, we use a metric called *variety* [10]. Holland ([10]) states that the complexity of the system behavior can be analyzed with respect to the measures of variety and *intensity of constraint*, where variety relates to the number of control states of a system to the number of variations in control to achieve effective response. The variety (V) and intensity of constraint (I) are given by Equation 33, where V_p and V_s are variety in an unconstrained and constrained system, respectively. m is possible number of states in each automata and M is the number of states exhibited (traversed) by the system. Holland states that increasing I (or decreasing V_s) interprets that system is tending towards Type IIa emergence, while decreasing I (or increasing V_s) interprets the system moving towards Type IIb emergent behavior.

$$I(t) = \frac{V_p}{V_s(t)} - 1 = \frac{N \log_2 m}{\log_2 M(t)} - 1 \tag{33}$$

Figure 33 illustrates an example for detection of the emergent property formation using the variety metric. As shown, when the groups are formed (undesirable state) in UAVs, variety becomes flat, whereas it fluctuates when no groups are formed (desirable state). The plots are obtained using MATLAB [68].

Here, we also want to compare the detection process by dimensional versus dimensionless variables (π's) explained in Section 3.3. Consider the case with two UAVs performing the persistent surveillance of the circular plume. The state vector for dimensional and dimensionless variables is given below:

$$State(Dimensional) = \begin{bmatrix} x_1(k) & y_1(k) & v_{x1}(k) & v_{y1}(k) \\ x_2(k) & y_2(k) & v_{x2}(k) & v_{y2}(k) \end{bmatrix} \tag{34}$$

$$State(Dimensionless) = \begin{bmatrix} x_1(k)/y_1(k) & v_{x1}(k)/v_{y1}(k) \\ x_2(k)/y_2(k) & v_{x2}(k)/v_{y2}(k) \end{bmatrix} \tag{35}$$

Figures 34 and 35 demonstrate the detection of emergence using the variety metric for both cases. In this simulation, emergence (aligned movement of agents - undesirable behavior) starts approximately at tick = 270. The plots show that both techniques can equally well detect the change in the variety at that tick location, but the variety value falls in case of dimensionless variables, making it easier to detect the emergence mathematically. Though both techniques perform well in terms of simulation, they differ in the magnitude in terms of the complexity (discussed in a later section).

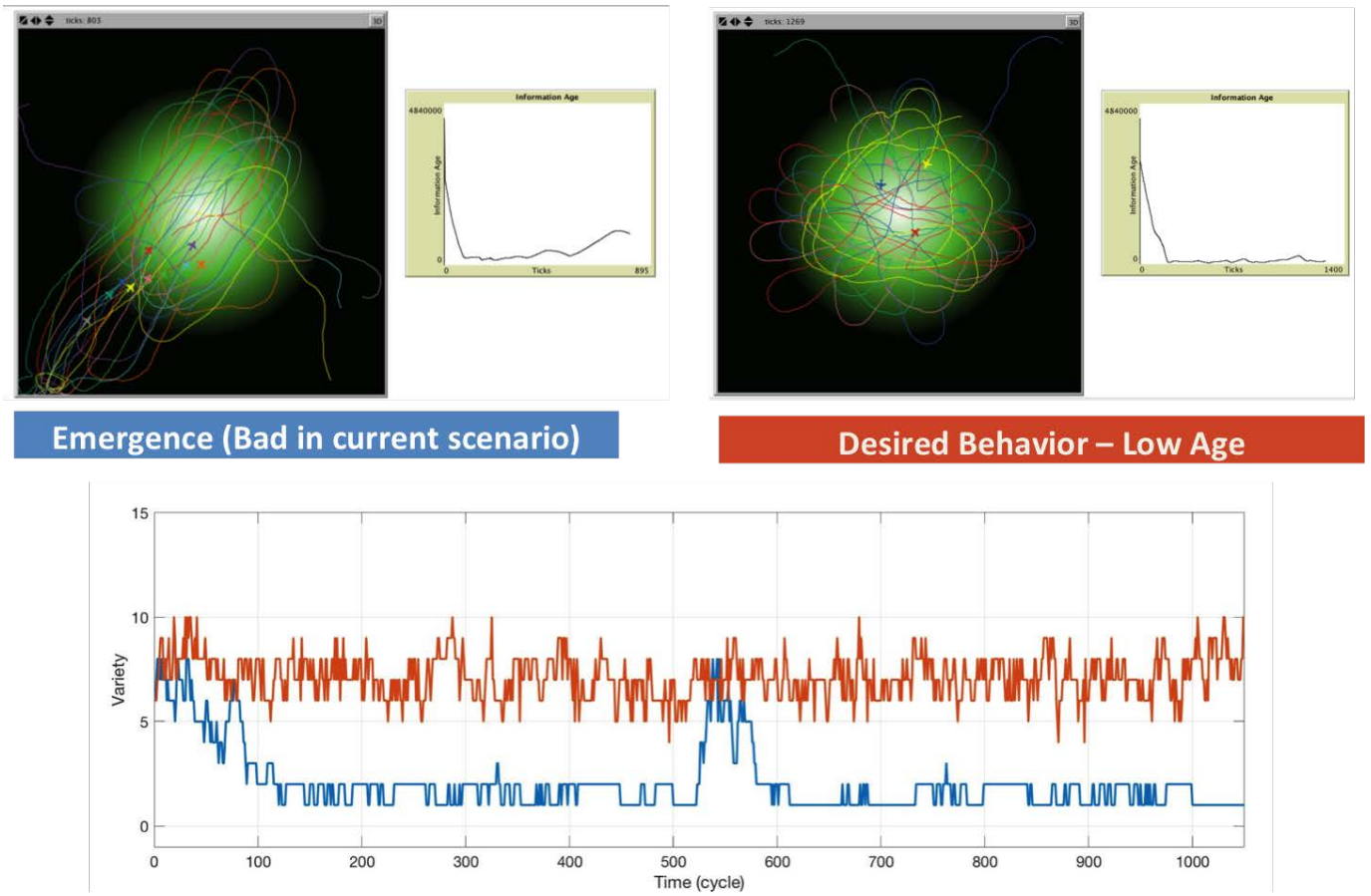


Figure 33. Detection of Emergent Behavior using Variable-Based Approach (Variety Metric)

4.4.2 Detection using FSM. As we mentioned before, the behavior of multi-agent systems can be represented by a FSM in our behavior ontology. So, we can recognize the behavior of a system with FSM representation using the following three steps:

- Given a sequence of data (e.g., event trace) observed from the simulation of a system's behavior
- Determine if this trace is compatible with the known $FSMs$ (i.e., if the input trace is a path in the known FSM graph)
- If the input trace is compatible with a known FSM^*, and this known FSM^* represents a certain behavior, recognize this system's behavior as compatible with FSM^*

We have two known FSMs as instances of FSM class: fsm1, fsm2 (Fig. 36). They represent the circular motion behavior shown in the test case scenario (Fig. 24). We represent these two FSMs in our ontology (Fig. 37).

Suppose there are two event traces (shown in Fig. 38). By running the inference engine to infer if each trace is compatible with any of the FSM instances described above, the inference results show that trace1 is compatible with both fsm1 and fsm2, and trace2 is compatible with fsm1.

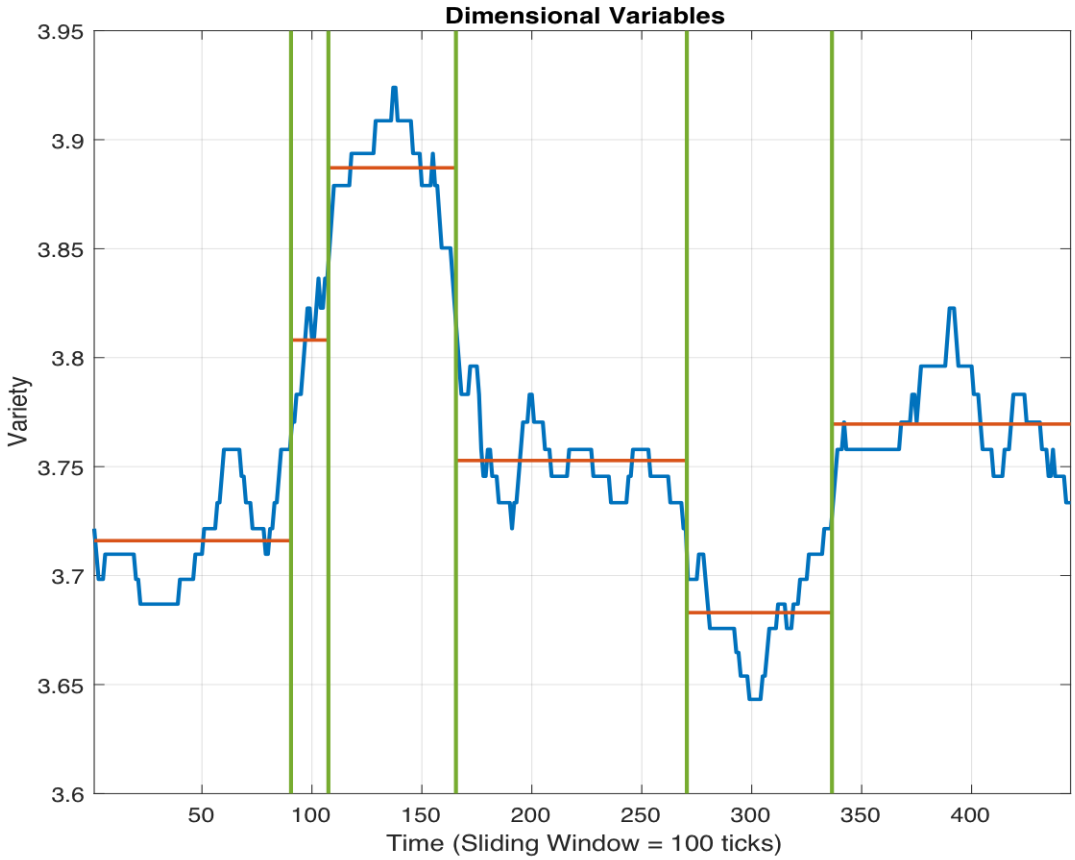


Figure 34. Detection using Variety with Dimensional Variables

4.4.3 Classification using Ontology Inference. Representing and classifying all possible cases of emergence in a system is also an important goal for our research. We achieve this goal using our behavior ontology and inference engine. In our behavior ontology, we have two classes to represent behaviors, one is Behavior class, which contains the actual behavior instances. Another one is Behavior Model class, which contains the models of different behavior types as subclasses. These behavior instances are imported from simulation, which usually described by some basic attributes, e.g., velocity, heading. Our goal is to infer which type of behavior model the behavior instances belong to. We list several specific types of behavior model, and define these models by OWL axioms. Once we import the attributes of the behavior instances, OWL reasoners infer the behavior model type for each behavior instance automatically.

In Figure 10, Flocking is one of the subclasses of Behavior Model class represented in our ontology. We explain the details of this behavior model using an example to show how the inference works. Basically, a UAV flocking scenario has the following rules:

- A UAV flocking should contain at least two UAVs.
- The UAVs in a flock should have the same velocity.

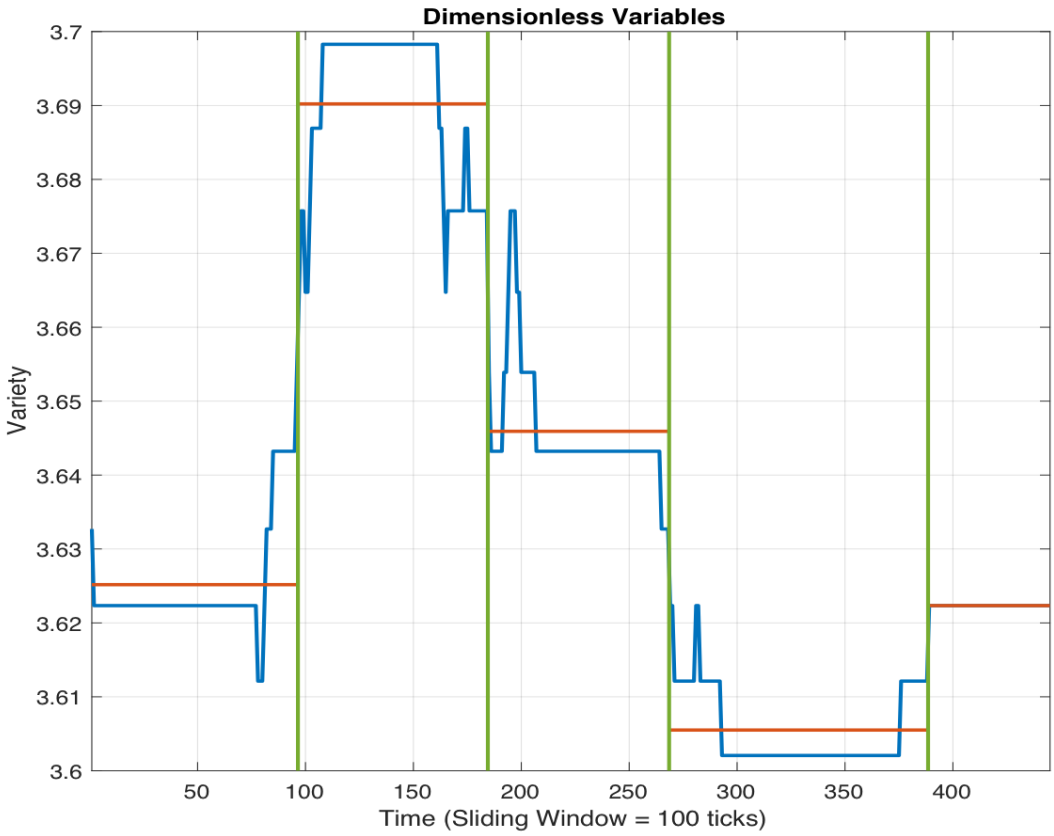


Figure 35. Detection using Variety with Dimensionless Variables

- The UAVs in a flock should have the same heading.
- The UAVs in a flock should be close to each other.

We use OWL axioms to define the UAV flocking class. Figure 39 shows the OWL axioms in Protégé.

The behavior instances should be imported from simulation automatically. But so far, in this example, we add a few behavior instances based on the simulation manually. We create a behavior instance named *"UAVSystemBehavior1"*, which has two participants: uav1 and uav2. We assign velocity value and heading value to each UAV. The values satisfied the rules described above, the OWL reasoner infers the *UAVSystemBehavior1* to be an instance of Flocking.

4.5 Learning Hypersurfaces

To make control decisions, we need knowledge the model of the system under consideration. The aim for the learning is to create a database with model's dynamics for every qualitatively distinct behavior. Such a database represents a partial model of the overall system's behavior. This is achieved by learning "hypersurfaces". Using similitude theory (dimensional analysis),

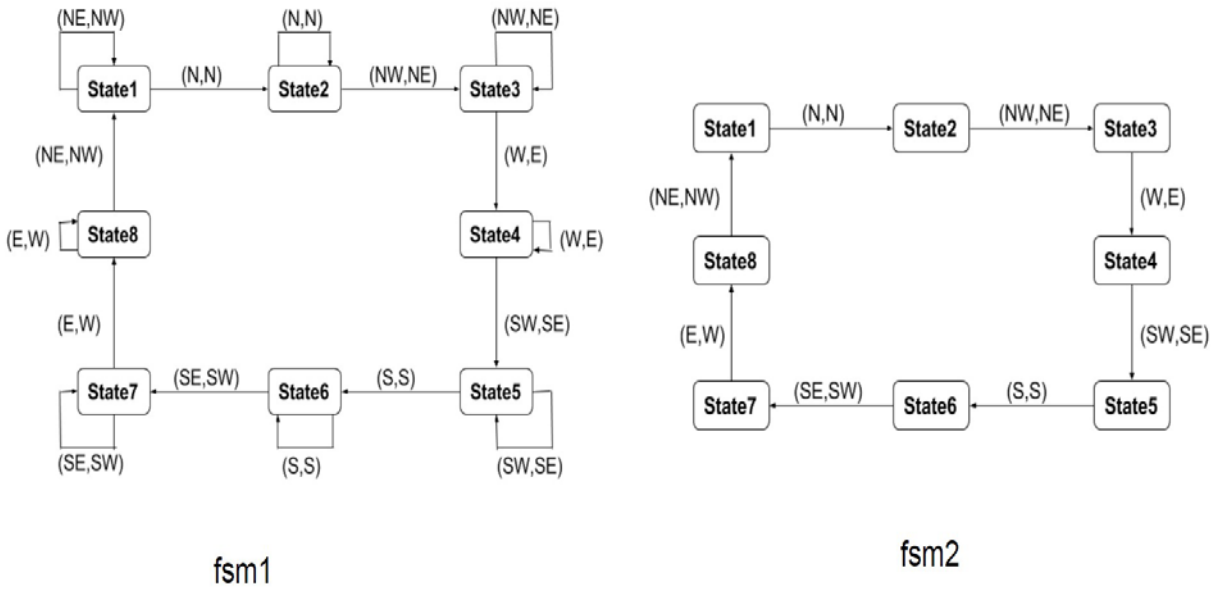


Figure 36. Two Instances of FSM

we subdivide the domain of state transition function into hypersurfaces (or orbits). Similarity property allows us to store only one point from each hypersurface, resulting in a reduced space complexity. Hypersurfaces are used for separating qualitatively different behaviors of the system. More specifically, the database points representing particular hypersurfaces are used for predicting of the next states. We use the hypersurface learning algorithm defined in [2]. The algorithm is presented in Algorithm 3 below. Figure 40 depicts the speed of learning of this algorithm in terms of the number of landmark points that the system stores. The straight line towards the end of the plot shows that the learning saturates. These results were obtained while learning hypersurfaces in the persistent surveillance scenario with two UAVs.

Hypersurfaces are learned using landmark points. We compared the performance of learning algorithm in the UAV scenario based on π's (dimensionless quantities) and the original variables. For instance, in a simulation, the number of distinct behaviors (landmark points) learned using the traditional approach is 90, while using the π's, this number was 46, resulting in almost 50% reduction is space. Figure 41 shows this comparison.

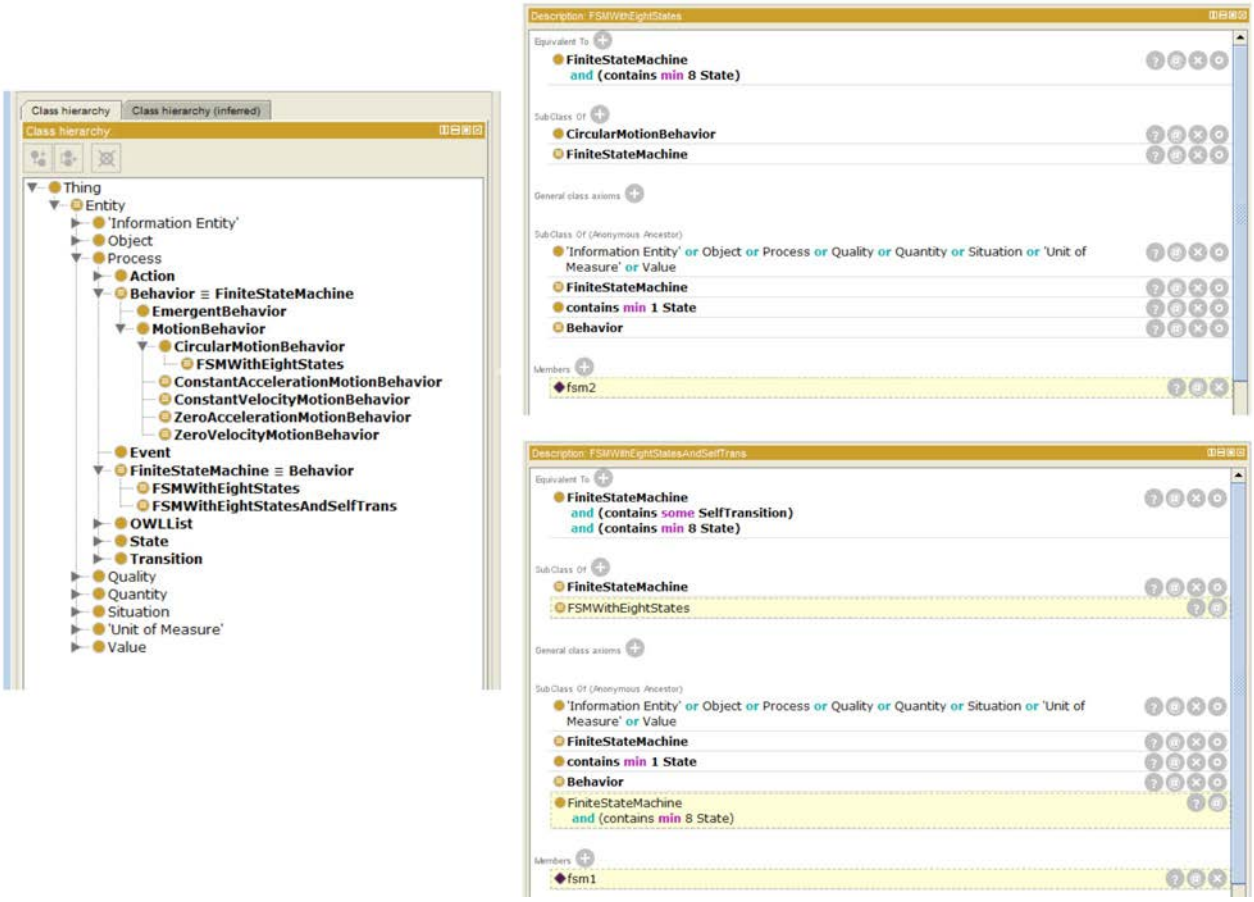


Figure 37. Representation of Two FSM Instances in OWL

Algorithm 3 Hypersurface Learning

1: Identify landmark points in output variable space
2: Calculate π terms using measured variables into the formula obtained using dimensional analysis (*critical hypersurface*)
3: Match derived hypersurface against already learned hypersurfaces
4: **if** matched **then**
5: Interpolate output value from the rest of the stored hypersurfaces in model database
6: If interpolation succeeds, remove redundant hypersurface
7: Else go to Step 1
8: **else**
9: Interpolate output value from hypersurfaces stored in database
10: If interpolated value differs from measured than a given threshold, save the derived hypersurface
11: Else go to Step 1
12: **end if**

trace1: (N,N) -> (NW,NE) -> (W,E) -> (SW,SE) -> (S,S) -> (SE,SW) -> (E,W) -> (NE NW) -> (N,N)

trace2: (N,N) -> (N,N) -> (NW,NE) -> (NW,NE) -> (W,E) -> (W,E) -> (SW,SE) -> (SW,SE) -> (S,S) -> (S,S) -> (SE,SW) -> (SE,SW) -> (E,W) -> (E,W) -> (NE NW) -> (NE NW) -> (N,N)

Figure 38. Two Event Trace Examples

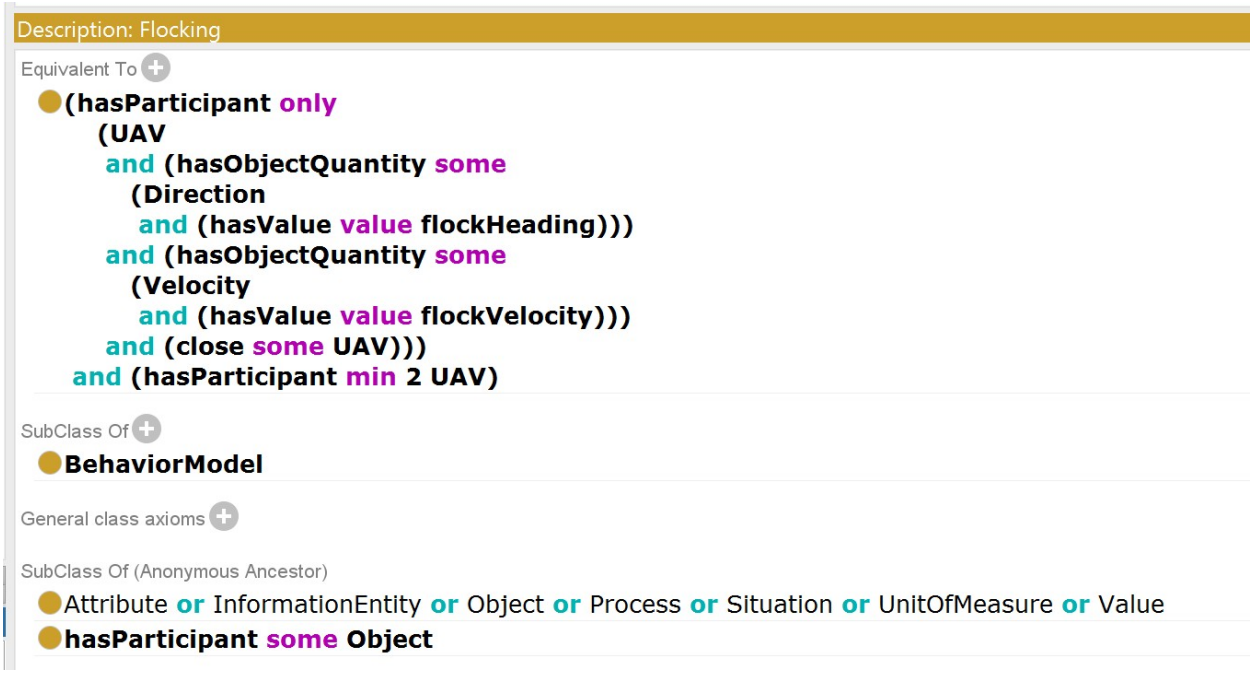


Figure 39. UAV Flocking Example

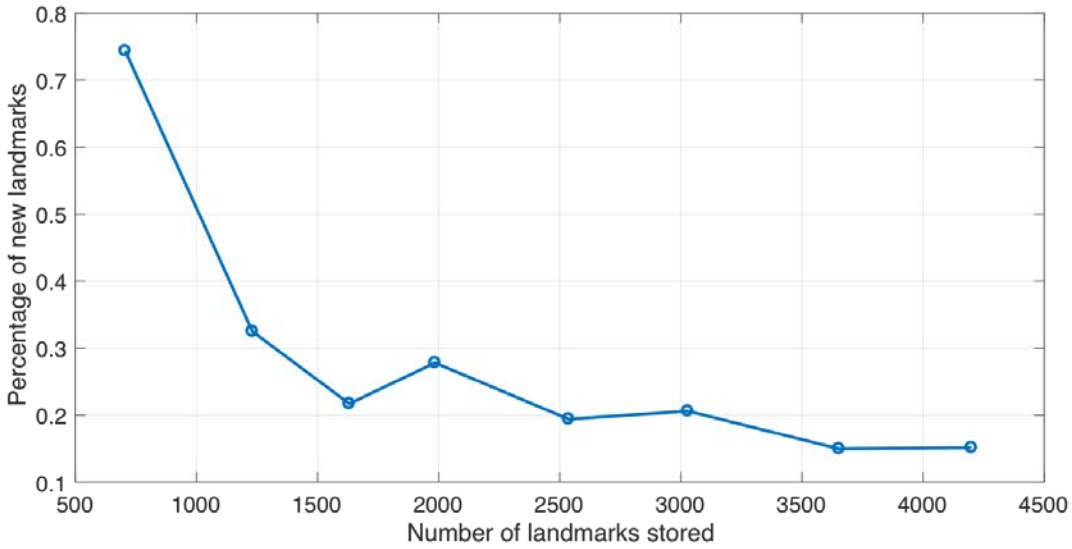


Figure 40. Accuracy of Learning Algorithm [2]

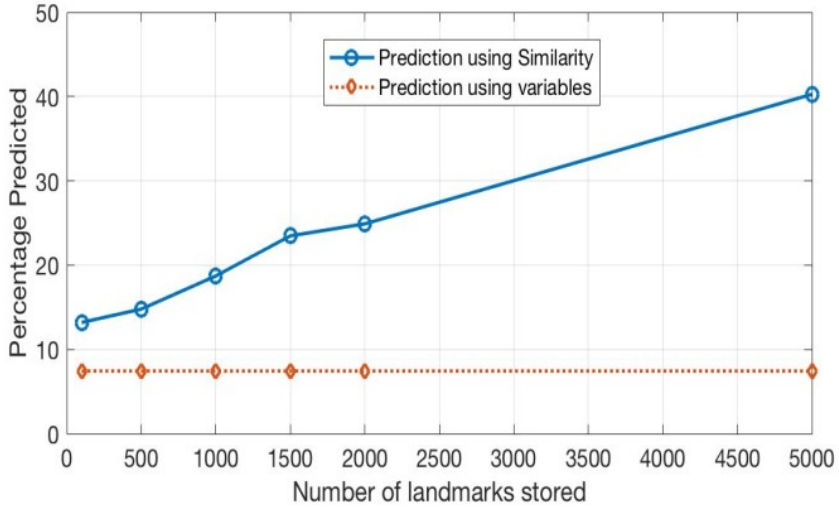


Figure 41. Comparison between Prediction by Similarity and Traditional Approach

4.6 Complexity Analysis

In this section, we discuss the complexity reduction achieved by using the dimensional analysis and Q^2 approach presented in this report. The reduction in complexity is obtained by reducing the state space by abstracting quantitative dynamical system to the qualitative dynamical system. Additionally, the decrease in the dimensionality of the system space $\Re^n$, reduced value of *n*, is achieved by considering the impact of couplings between various dynamical systems, i.e., by using dimensionless quantities instead of directly using the dimensional variables.

1. *Impact of qualitative abstraction on complexity of behavior recognition*:
   Section 4.4.2 discussed the detection of emergent behaviors using FSMs by matching the traces to the stored automata. Here we analyze the impact of using qualitative approach on the complexity of behavior recognition using FSMs. To recognize the behavior, the reasoner needs to match traces to possible automata. For calculating the complexity analysis, we need to calculate the number of possible traces for a given number of transitions. For *N* partitions (qualitative regions) of the quantitative space, an agent can go through *N* transitions with the total number of traces (*T)* given by:

$$T = 1! + 2! + 3! + ... + N! \tag{36}$$

   For instance, with two transitions, i.e., $\{t_1, t_2\}$, four traces can be generated, i.e., $\{t_1, t_2, t_1t_2, t_2t_1\}$ where the possible automata for each trace are shown in Figure 42. Figure 43 demonstrates the number of comparisons performed by the reasoner as the number of qualitative partitions of qualitative regions increases. Here the complexity is on the order of $O(T^2)$, where *T* is the number of traces.

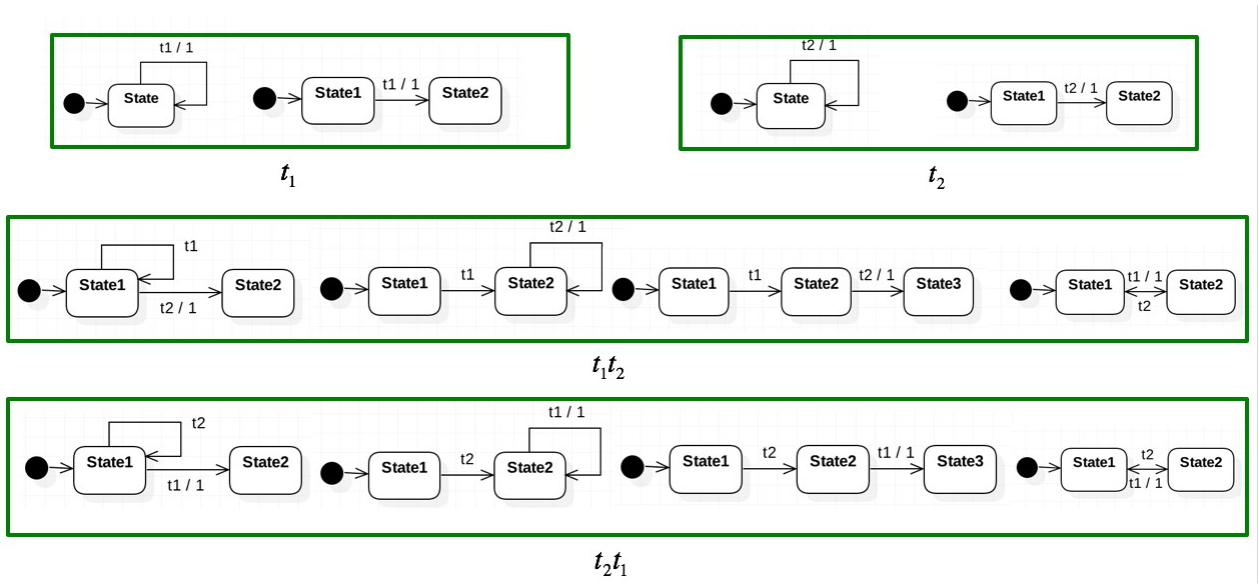


Figure 42. Possible Automata for Four Traces

2. *Impact of using dimensional versus dimensionless approaches on size of the state space*:

   Let z be the number of possible states a single UAV can take and N be the total number of UAVs in a system. With dimensional variables, the state space increases exponentially as N increase, i.e., the complexity is in the order $O(z^N)$. On the other hand, with dimensionless variables, the increase in the state space is just quadratic, i.e., ($O(z^2)$) complexity, which provides a lot of improvement over the traditional approach. Figure 44 gives the number of states with dimensional and dimensionless variables as N increases. The figure demonstrates the exponential increase in the number of states as the number of UAVs increases in the case of dimensional variables.

3. *GDS versus QDS:*

   We also compare the performance of General Dynamical System (GDS) versus Qualitative Dynamical System (QDS) in terms of the number of comparisons. We observed that GDS is an order of magnitude more computationally expensive than QDS. Figure 45 shows this comparison. The plots show that for the same number of ticks (or iterations), GDS perform 140 comparisons while QDS performs just 14.

5 CONCLUSION

This report summarizes our efforts to investigate an approach to modeling and analysis of multi-agent dynamical systems with the intent of reduction in the complexity of detecting and controlling undesirable emergent behaviors. The approach is a combination of dimensional analysis (similitude theory) and the Quantitative-Qualitative (Q^2) approach for representing dynamical systems. We presented a number of experiments that were performed in this work using multi-UAV systems which were performing a specified mission of

persistent surveillance of targeted search area (Plume Monitoring). The experiments showed different types and forms of emergence in swarm(s) of UAVs. The approach used in our work was able to detect undesirable emergent behaviors which are not beneficial for the scenario discussed. We also showed how the qualitative approach is computationally inexpensive as compared to the traditional approaches that use quantitative representations of the dynamical systems. The complexity reduction is primarily due to the reduction of the state space by abstracting quantitative dynamical systems to qualitative dynamical systems. Additionally, experiments showed that the use of similitude theory (supported by machine learning) lead to the improvements in the efficiency of detection of emergent behaviors.

Partitions (N)	Traces (T)	Automata	Comparisons
2	2	8	4
3	6	36	36
4	24	192	576
5	120	1200	14400
6	720	8640	518400
7	5040	70560	25401600
8	40320	645120	1625702400
9	362880	6531840	131681894400
10	3628800	72576000	13168189440000
	$N!$	$2 \times T \times len(T)$	T^2

Figure 43. Number of Comparisons for *N* Qualitative Partitions

Another aspect of our research was to gain more understanding of the nature of emergence in general. Towards this aim, we performed an intensive study of the definitions of emergence. As a result, we identified very many definitions of this concept. Then we identified a number of features of the particular definitions and performed Formal Concept Analysis. The net result was a lattice that shows the dependencies between and among the various definitions. The first conclusion from this research was that there is a lack of agreement on the notion of emergence. The encouraging result was that many of the definitions have common parts. Unfortunately, many of the definitions, even though they use the same linguistic terms, it is not clear what the meaning of the terms is. In particular, the most controversial, although very crucial, term is "irreducibility". Unfortunately, none of the definitions we reviewed are based in mathematics. While this is not the only one way of stating definitions, another one being *precise* definitions, expressed in natural language. We have identified several precise definitions of irreducibility in mathematics. However, while they can serve as exemplars, none of them is a good match for the purpose of defining this term for identifying emergence. In addition to mathematics, we have also analyzed computational irreducibility, where it applies

to the computational process. Since emergence is also is associated with processes, computational irreducibility is perhaps the best match to our needs. The problem is that this definition also is not fully precise. It refers to a precise term used in both mathematics and computer science - the undecidability - but the definition does not provide a well-defined link between undecidability and emergence. In summary, even though we have not found a fully satisfactory definition of emergence, we have identified a potential direction for the future research of this concept.

Number of UAVs (N)	Dimensional Variables	Dimensionless Variables
2	144	144
3	1728	288
4	20736	432
5	248832	576
6	2985984	720
7	35831808	864
8	429981696	1008
9	5.1598e+09	1152
10	6.1917e+10	1296
20	3.8338e+21	2736
	z^N	$(N-1)z^2$

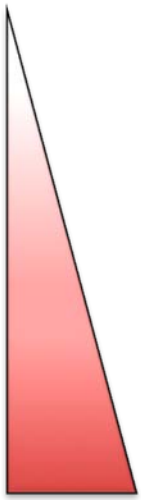


Figure 44. Comparison of Number of States with Dimensional and Dimensionless Variables (z = 12)

5.1 Future Direction

Emergence is one of the fundamental properties of complex systems and in recent years it has captured significant attention from the scientific community. Emergence appears in different forms and shapes in a variety of systems from simple to the most complex. Emergence can be viewed as a positive as well as a negative phenomenon, i.e., it can either significantly improve the functional performance of an engineered system, or it can have a very negative impact on the system's functionality and performance. Emergence is a territory which is not properly defined in the literature, yet. Thus, there is a need for a mechanism that provides a structured approach to the analysis and control of such behaviors. We address this issue by proposing a framework for exploration of emergent behaviors in multi-agent systems.

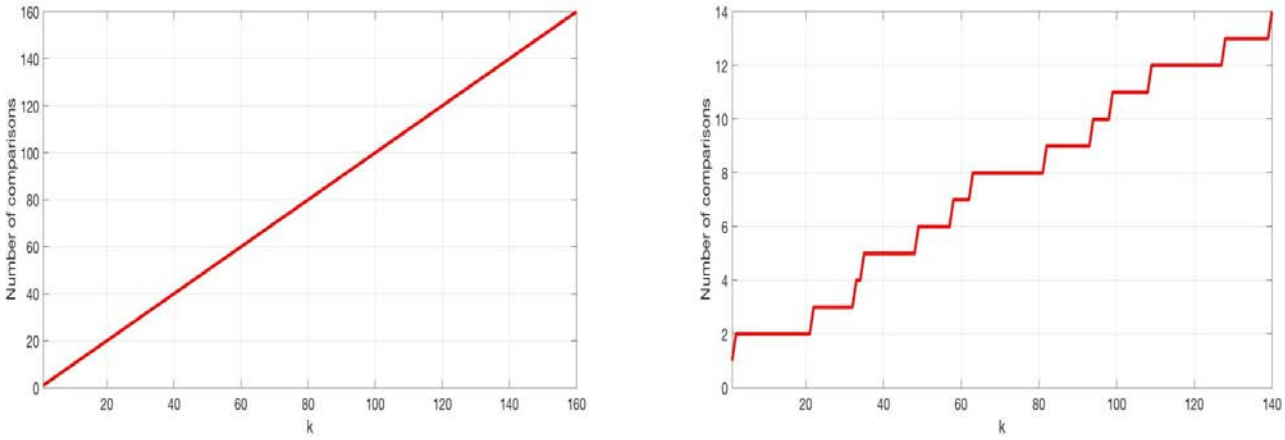


Figure 45. General Dynamical System - GDS (Left) versus Qualitative Dynamical System - QDS (Right)

This research is still work-in-progress. Our ultimate aim here is to design a mechanism to control undesirable behaviors in CAS (specifically swarms of UAVs). We presented our partial work in [69, 46]. Our published work and papers in preparation are listed in Appendix A. Some of the existing questions in the field of emergence that are targeted in our research are: Can we develop a formal model of emergent behaviors? Can we do detection of unexpected undesirable behaviors in an efficient and reliable way? Since analyzing features of multi-agent system suffers from a high computational complexity due to a high dimension of the analysis space, can we find an approach to reduce the complexity of analysis? Can we create a complete taxonomy of emergence?

Acknowledgement: This research was conducted in close collaboration with Dr. Paul Kogut from Lockheed Martin who was sponsored by a separate project from the Air Force Laboratory.

References

[1] J. Fromm, "Types and forms of emergence," *arXiv preprint nlin/0506028*, 2005.

[2] M. M. Kokar and J. Reeves, "Qualitative monitoring of time-varying physical systems," in *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*. IEEE, 1990, pp. 1504–1508.

[3] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH computer graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[4] C. S. Holling, "Understanding the complexity of economic, ecological, and social systems," *Ecosystems*, vol. 4, no. 5, pp. 390–405, 2001.

[5] L. A. Adamic and B. A. Huberman, "Power-law distribution of the world wide web," *Science*, vol. 287, no. 5461, pp. 2115–2115, 2000.

[6] N. O'Neill, "Google now indexes 620 million facebook groups," *Web log post. All Facebook*, vol. 1, 2010.

[7] W. Cirne, F. Brasileiro, D. Paranhos, L. F. W. Góes, and W. Voorsluys, "On the efficacy, efficiency and emergent behavior of task replication in large distributed systems," *Parallel Computing*, vol. 33, no. 3, pp. 213–234, 2007.

[8] S. Kaisler and G. Madey, "Complex adaptive systems: Emergence and self-organization," *Tutorial Presented at HICSS-42 Big Island*, 2009.

[9] W. K. V. Chan, "Interaction metric of emergent behaviors in agent-based simulation," in *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference, 2011, pp. 357–368.

[10] O. T. Holland, "Taxonomy for the modeling and simulation of emergent behavior systems," in *Proceedings of the 2007 spring simulation multiconference-Volume 2*. Society for Computer Simulation International, 2007, pp. 28–35.

[11] C. Szabo and Y. M. Teo, "Post-mortem analysis of emergent behavior in complex simulation models," in *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, 2013, pp. 241–252.

[12] G. H. Lewes, Problems of life and mind. Trübner & Company, 1877.

[13] D.Fisher, "An emergent perspective on interoperation in systems of systems," 2006.

[14] C. Szabo and Y. M. Teo, "Semantic validation of emergent properties in component-based simulation models," in Ontology, Epistemology, and Teleology for Modeling and Simulation. Springer, 2013, pp. 319–333.

[15] G. E. Marsh, "The demystification of emergent behavior," arXiv preprint *arXiv:0907.1117*, 2009.

[16] J.P.Müller, "Emergence of collective behaviour and problem solving," in *International Workshop on Engineering Societies in the Agents World*. Springer, 2003, pp. 1–20.

[17] I. Allison and Y. Merali, "Software process improvement as emergent change: A structurational analysis," *Information and software technology*, vol. 49, no. 6, pp. 668–681, 2007.

[18] J. C. Mogul, "Emergent (mis)behavior vs. complex software systems," in *IN EUROSYS*. ACM, 2006, pp. 293–304.

[19] C. W. Johnson, "What are emergent properties and how do they affect the engineering of complex systems?" *Reliability Engineering & System Safety*, vol. 91, no. 12, pp. 1475–1481, 2006.

[20] J. Deguet, Y. Demazeau, and L. Magnin, "Elements about the emergence issue: A survey of emergence definitions," *ComPlexUs*, vol. 3, no. 1-3, pp. 24–31, 2006.

[21] D. Chalmers, "The conscious mind, new york: Oxford univ," 1996.

[22] G. B. Dyson, *Darwin among the machines: The evolution of global intelligence*. Basic Books, 2012.

[23] J. Holland and H. Mallot, "Emergence: from chaos to order," *Nature*, vol. 395, no. 6700, pp. 342–342, 1998.

[24] J. Goldstein, "Emergence as a construct: History and issues," *Emergence*, vol. 1, no. 1, pp. 49–72, 1999.

[25] E.O'Toole, V.Nallur, and S. Clarke, "Towards decentralised detection of emergence in complex adaptive systems," in *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 2014, pp. 60–69.

[26] Wikipedia, "Formal Concept Analysis — Wikipedia, the free encyclopedia," https://en. wikipedia.org/wiki/Formal_concept_analysis, 2017, [Online; accessed 3-August-2017].

[27] J. P. Crutchfield, "Is anything ever new?-considering emergence," in Citeseer, 1994.

[28] J. Kim, "Making sense of emergence," *Philosophical studies*, vol. 95, no. 1, pp. 3–36, 1999.

[29] C. Rouff, A. Vanderbilt, M. Hinchey, W. Truszkowski, and J. Rash, "Properties of a formal method for prediction of emergent behaviors in swarm-based systems," in *Software Engineering and Formal Methods, 2004. SEFM 2004. Proceedings of the Second International Conference on*. IEEE, 2004, pp. 24–33.

[30] F. Patterson Jr, "Life cycles for system acquisition," *Systems Engineering and management for Sustainable Development-Volume I*, p. 82, 2009.

[31] J. C. Hsu and M. Butterfield, "Emergent behavior of systems-of-systems," *Proceedings of Conference INCOSE-2009*, pp. 1–27, 2009.

[32] C. J. Alberts, A. J. Dorofee, R. Creel, R. J. Ellison, and C. Woody, "A systemic approach for assessing software supply-chain risk," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on*. IEEE, 2011, pp. 1–8.

[33] V. Kim, "A design space exploration method for identifying emergent behavior in complex systems," Ph.D. dissertation, Georgia Institute of Technology, 2016.

[34] S. Wolfram, *A New Kind of Science*. Wolfram Media, Inc., 2002.

[35] D. J. Chalmers, "Strong and weak emergence," *The reemergence of emergence*, pp. 244–256, 2006.

[36] M. Bedau, "Downward causation and the autonomy of weak emergence," *Principia*, vol. 6, no. 1, p. 5, 2002.

[37] W. Seager, "Emergence, epiphenomenalism and consciousness," *Journal of Conscious- ness Studies*, vol. 13, no. 1-2, pp. 21–38, 2006.

[38] L. B. Rainey and A. Tolk, *Modeling and simulation support for system of systems engineering applications*. John Wiley & Sons, 2015.

[39] R. Gore and P. F. Reynolds Jr, "An exploration-based taxonomy for emergent behavior analysis in simulations," in *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*. IEEE Press, 2007, pp. 1232–1240.

[40] Y. Bar-Yam, "Multiscale variety in complex systems," *Complexity*, vol. 9, no. 4, pp. 37–45, 2004.

[41] Y. M. Teo, B. L. Luong, and C. Szabo, "Formalization of emergence in multi-agent systems," in *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, 2013, pp. 231–240.

[42] A. G. Madey and G. R. Madey, "Design and evaluation of uav swarm command and control strategies," in *Proceedings of the Agent-Directed Simulation Symposium*. Society for Computer Simulation International, 2013, p. 7.

[43] E. Bonabeau, J.-L. Dessalles, and A. Grumbach, "Characterizing emergent phenomena (1): A critical review," *Revue internationale de systémique*, vol. 9, no. 3, pp. 327–346, 1995.

[44] C. Emmeche, S. Køppe, F. Stjernfelt *et al.*, "Levels, emergence, and three versions of downward causation," *Downward causation. Minds, bodies and matter*, pp. 13–34, 2000.

[45] O. T. Holland, *Partitioning method for emergent behavior systems modeled by agent-based simulations*. Old Dominion University, 2012.

[46] S. Singh, S. Lu, M. M. Kokar, and P. A. Kogut, "Detection and classification of emergent behaviors using multi-agent simulation framework (wip)," in *Proceedings of the MSCIAAS, SpringSim*, Society for Computer Simulation International, 2017.

[47] S. Bouarfa, H. A. Blom, R. Curran, and M. H. Everdij, "Agent-based modeling and simulation of emergent behavior in air transportation," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, p. 1, 2013.

[48] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in *International Workshop on Agent Theories, Architectures, and Languages*. Springer, 1996, pp. 21–35.

[49] Wikipedia, "OODA loop — Wikipedia, the free encyclopedia," https://en.wikipedia.org/wiki/OODA_loop, 2017, [Online; accessed 2-January-2017].

[50] S. J. Taylor, "Introducing agent-based modeling and simulation," in *Agent-based Modeling and Simulation*. Springer, 2014, pp. 1–10.

[51] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation," in *Proceedings of the 37th conference on Winter simulation*. Winter Simulation Conference, 2005, pp. 2–15.

[52] E. M. Ronald, M. Sipper, and M. S. Capcarrère, "Design, observation, surprise! a test of emergence," *Artificial Life*, vol. 5, no. 3, pp. 225–239, 1999.

[53] E. Bonabeau, J.-L. Dessalles, and A. Grumbach, "Characterizing emergent phenomena (2): a conceptual framework," *Revue Internationale de Systémique*, vol. 9, no. 3, pp. 347–371, 1995.

[54] S. Christley, X. Xiang, and G. Madey, "An ontology for agent-based modeling and simulation," in *Proceedings of the agent 2004 conference*. Citeseer, 2004.

[55] M. Moshirpour, R. Alhajj, M. Moussavi, and B. H. Far, "Detecting emergent behavior in distributed systems using an ontology based methodology," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2407–2412.

[56] Y. M. Teo and C. Szabo, "Codes: An integrated approach to composable modeling and simulation," in *41st Annual Simulation Symposium (anss-41 2008)*. IEEE, 2008, pp. 103–110.

[57] W3C, "Owl web ontology language overview." [Online]. Available: http://www.w3.org/ TR/owl-features/

[58] T. Gruber, "Ontology," *Encyclopedia of database systems*, pp. 1963–1965, 2009.

[59] S. F. Railsback, S. L. Lytinen, and S. K. Jackson, "Agent-based simulation platforms: Review and development recommendations," *Simulation*, vol. 82, no. 9, pp. 609–623, 2006.

[60] S. Tisue and U. Wilensky, "Netlogo: Design and implementation of a multi-agent modeling environment," in *Proceedings of agent*, vol. 2004, 2004, pp. 7–9.

[61] E. Buckingham, "On physically similar systems; illustrations of the use of dimensional equations," *Physical review*, vol. 4, no. 4, p. 345, 1914.

[62] S. J. Kline, *Similitude and approximation theory*. Springer Science & Business Media, 2012.

[63] M. M. Kokar, "Critical hypersurfaces and the quantity space." in *AAAI*, 1987, pp. 616– 620.

[64] M. Kokar and S. Reveliotis, "Integrating qualitative and quantitative methods for model validation and monitoring," in *Intelligent Control, 1991., Proceedings of the 1991 IEEE International Symposium on*. IEEE, 1991, pp. 286–291.

[65] M. M. Kokar, "Learning to select a model in a changing world," in *Machine Learning Proceedings 1991: Proceedings of the Eighth International Workshop (ML91)*. Morgan Kaufmann, 1991, p. 313.

[66] S. A. Reveliotis and M. M. Kokar, "A framework for on-line learning of plant models and control policies for restructurable control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 11, pp. 1502–1512, 1995.

[67] M. M. Kokar, "On consistent symbolic representations of general dynamic systems," *IEEE transactions on systems, man, and cybernetics*, vol. 25, no. 8, pp. 1231–1242, 1995.

[68] MATLAB, *version 9.1 (R2016b)*. Natick, Massachusetts: The MathWorks Inc., 2016.

[69] P. Kogut, M. Kokar, S. Singh, and S. Lu, "Detecting undesirable emergent behavior in teams of autonomous uass," in *Safe and Secure Systems and Software Symposium (S5)*, 2016.

APPENDIX A - Publications and Presentations

A.1 Published Work

1. Singh, S., Lu, S., Kokar, M., and Kogut, P., "Detection and Classification of Emergent Behavior using Multi-Agent Simulation Framework (WIP)". Spring Simulation Multi-Conference (MSCIAAS), 2017.

2. Kogut, P., Kokar, M., Singh, S., and Lu, S. 2016. "Detecting Undesirable Emergent Behavior in Teams of Autonomous UASs". In Safe and Secure Systems and Software Symposium (S5), 2016.

A.2 Papers in Preparation

1. Singh, S., "Detection of Emergent Behaviors in System of Dynamical Systems Using Similitude Theory", Ph.D. Colloquium, Winter Simulation Conference, 2017. *(Extended Abstract Submitted)*

2. Singh, S., Lu, S., Kokar, M., and Kogut, P., "Experiments with Detection and Classification of Emergent Behavior using Multi-Agent Simulation Framework".

3. Singh, S., Lu, S., Kokar, M., and Kogut, P., "Detection of Emergent Behaviors in Swarms of UAVs Monitoring Chemical Plume Based on Similitude Theory".

4. Singh, S., Lu, S., Kokar, M., and Kogut, P., "Detection of Emergent Behaviors in Swarms of UAVs Monitoring Chemical Plume Based on Q2 Approach".

5. Singh, S., Lu, S., Kokar, M., and Kogut, P., "Ontological Classification of Emergent Behaviors".

APPENDIX B - Abstracts

B.1 Spring Simulation Multi-Conference, 2017

In recent years, the concept of emergence has gained much attention in the field of complex systems. However, the inability to predict and control emergent phenomena prevents us from exploring its full potential. The research effort in this paper focuses on exploring emergent behaviors by proposing a framework for analysis of systems that exhibit such behaviors. The framework provides a platform for simulating and analyzing behaviors in multi-agent system, including detection and classification of emergence into different types. In this paper, we follow the classification of emergent behaviors according to Fromm's taxonomy. In addition, the paper presents a scenario implementation using swarms of Unmanned Aerial Vehicles (UAVs) to demonstrate the applicability of the proposed approach. Since this is a part of on-going research, future direction is also discussed.

B.2 Safe and Secure Systems and Software Symposium (S5), 2016

The Air Force envisions the use of cooperating teams of small unmanned aerial systems (UAS) for ISR and other missions instead of single large expensive UASs. There is also a trend toward increased autonomy of UASs for effective operation in a limited communications environment. The combination of cooperating teams and increased autonomy presents a major new challenge for test, evaluation, verification, and validation. We need to focus on interactions between UASs which involves emergent behavior and complex adaptive systems. There has been a lot of theoretical research on emergent behavior but how can we leverage this for the engineering of reliable systems of autonomous systems?

This talk will discuss an ongoing exploratory research project in the AFRL Trusted Autonomy program. In this project, we are attempting to define the problem space for undesirable emergent behavior in UAS teams. This includes developing an ontology of emergent behaviors and defining representative scenarios where undesirable behavior is possible. The scenarios focus on common tasks in ISR missions such as search, persistent surveillance and dynamic task allocation. We are developing machine learning techniques to detect emergent behavior early so that control policies can be adjusted before safety problems or major performance degradations can occur. The behavior of the team of UASs is defined in terms of a dynamical system. The goal is to learn models for the phase transition boundaries in the dynamical system. Simulations and complexity analysis are being developed to evaluate the detection techniques.

LIST OF ACRONYMS

CAS	Complex Adaptive Systems.
CGS	Centimeter–Gram–Second.
EBS	Emergent Behavior System.
FCA	Formal Concept Analysis.
FSM	Finite State Machine.
GDS	General Dynamical System.
ISR	Intelligence, Surveillance, and Reconnaissance.
OODA	Observe, Orient, Decide, Act.
OWL	Web Ontology Language.
QDS	Qualitative Dynamical System.
QSM	Qualitative State Machine.
Re	Reynolds Number.
SI	Système Internationale.
UAV	Unmanned Aerial Vehicle.
$\boldsymbol{Q^2}$	Quantitative-Qualitative.